

NEURAL INFORMATION STORAGE AND PROCESSING IN THE
PRESENCE OF NOISE AND NONLINEARITY

WALEED EL-GERESY

Thesis submitted for the degree of Doctor of Philosophy

Supervised by PROFESSOR DENİZ GÜNDÜZ
Co-supervised by DR CHRISTOS PAPAVASSILIOU
Information Processing and Communications Lab
Department of Electrical and Electronic Engineering
Imperial College London

March 2024

COPYRIGHT DECLARATION

The copyright of this thesis rests with the author. Unless otherwise indicated, its contents are licensed under a [Creative Commons Attribution-NonCommercial 4.0 International Licence \(CC BY-NC\)](#).

Under this licence, you may copy and redistribute the material in any medium or format. You may also create and distribute modified versions of the work. This is on the condition that: you credit the author and do not use it, or any derivative works, for a commercial purpose.

When reusing or sharing this work, ensure you make the licence terms clear to others by naming the licence and linking to the licence text. Where a work has been adapted, you should indicate that the work has been changed and describe those changes.

Please seek permission from the copyright holder for uses of this work that are not included in this licence or permitted under UK Copyright Law.

DECLARATION OF ORIGINALITY

I hereby declare that the contents of this thesis, including the work and ideas detailed throughout, are solely the products of my own efforts and thought, except where otherwise appropriately referenced or credited.

London, March 2024

Waleed El-Geresy

ABSTRACT

Artificial intelligence has grown exponentially in popularity in recent years, thanks to impressive applications, especially in the field of generative modelling in visual and linguistic domains. A multitude of advancements in understanding of the theory of intelligence and of artificial neural networks have been made, which have drawn modern neural network architectures and algorithms further and further from their origins as models of networks of biological neurons in the human brain. This thesis argues that since the human brain is the source of our definition of intelligence, furthering our understanding of its computational methods will be the key to unbridling the field of AI from the reins of its huge energy costs, and bridging the gap between algorithms and the physical world. To this end, we explore a number of computational primitives present in biological neurons and neural networks, looking at the benefits, challenges and limitations of implementing them artificially, and arguing the case for an approach to AI research with awareness and exploitation of noise and non-ideality as fundamental components. To this end, we take a particular focus on devices called memristors, which have been proposed as passive electronic components that could be used for neuro-inspired computing. We investigate the ability of memristors to implement neuronal computational primitives, and address the problem of noisy information storage on the devices in the presence of resistive drift noise, using joint source-channel coding. We further explore the notion of intelligent processing in the presence of noise through identifying a use case for adversarial noise, generated as part of a competitive minimax communication game, exploring how it can be used to enable distribution matching in a generative modelling problem.

ACKNOWLEDGEMENTS

The journey from the inception to the creation of this thesis has been, like the natural world, highly non-linear and unpredictable. However, it seems that I am nearing a stable point, a clearing in the midst of a dense forest, a moment of clarity for the collection of thoughts. I would like to deeply thank my supervisors, Professor Deniz Gündüz and Dr Christos Papavassiliou, for their help and support throughout this long journey. I am also grateful to my examiners, Professor Danilo Mandic and Professor Tolga Duman, for putting in the time and effort to review my thesis and for their comments during my examination. I also want to express my gratitude to the members of IPC Lab, and all those I have met and worked with throughout my time at university, who have been supportive and inspirational.

I would like to thank my family, especially my mother Alia, my father Baher, my sister Hanya, my cousin Ahmed, and my dog Cwtch, who have helped me weather the toughest of storms. To mention everyone who has been with me throughout this journey would be impossible, but I would also like to thank some of those friends and colleagues who have been particularly supportive - Justin, Benedikt, Pu, Kartik, Haaris, Paul, Mahamood, Marian, Ahmed, Mehraj, Noel, Mike, Tim, Corey, Alex, Piers, Dan, Amirsalar, Jack, Dániel, Szymon, Kate, Burak, Adil, and Eric. Without you, I would not have reached this point.

And yet, it still cannot be said that this thesis would have been possible for me to complete alone, even with the support of those around me. This is because behind every creative leap, advance, and new insight or perspective, looms a huge body of human knowledge and experimentation stretching back not just weeks, months, or years, but centuries and millennia. I, like all humans on this planet, am a product of my upbringing and environment, and I am immensely fortunate to be alive at such an exciting time in history - a mere blip on cosmic scales. Yet somehow, I have managed to be born and brought up at the cusp of the onset of a second epoch of emergent self-awareness for human beings, where we become not only aware of our own minds, but also begin to understand how they might work and, indeed, be built: the recent boom in the field of artificial intelligence that will no doubt be a defining moment in the history of humankind. I am blessed to be able to exist at this time of intense curiosity and self-discovery; an era which the universe has possibly not seen until now.

CONTENTS

1	Introduction	25
1.1	What is Intelligence?	26
1.2	Accuracy and Energy	27
1.3	Computational Primitives	28
1.4	Neuromorphic Computing	29
1.5	Research Questions and Outline	30
1.6	Contributions	31
1.7	Publications	33
2	Background	35
2.1	Neurons and their Computational Primitives	36
2.1.1	Time Encoding of Information using Spikes	37
2.1.2	Spiking Nonlinearity	37
2.1.3	Synaptic Weight Strengthening and Decay	38
2.1.4	Learning Rules: Spike Timing Dependent Plasticity and Frequency Dependent Potentiation	38
2.1.5	Hebbian Competition and Lateral Inhibition	38
2.2	Memristors	39
2.2.1	Overview and History	39
2.2.2	Types of Memristor	41
2.2.3	Noise and Nonlinearity in Memristive Devices	43
2.2.4	Neuromorphic Computing with Memristors	45
2.3	Memristor Modelling	46
2.3.1	Modelling Considerations	46
2.3.2	Modelling Nonideality and Stochasticity	48
2.3.3	Data Driven Approaches	49
2.4	Source and Channel Coding	49
2.4.1	Source and Channel Coding	50
2.4.2	Information Theoretic Approaches for Addressing Nonidealities in Memristive Devices	53
2.4.3	Communication Games and Worst Case Noise	54
2.5	Generative Models	55
2.5.1	Generative Models as Parameterised Distributions	56
2.5.2	Types of Generative Model	56
2.6	The Use of Noise in Deep Learning	59
2.7	Summary	60
3	Event-Based Simulation of Stochastic Memristive Systems for Neuromorphic Computing	61
3.1	Introduction	61
3.1.1	Outline of Chapter	63

3.2	Event-Based Metastable Switching Model	64
3.2.1	Limitations of the Discrete-Time Stochastic Switching Models	64
3.2.2	State Modelling	66
3.2.3	Readout Equation	67
3.2.4	Switching Mechanism	69
3.2.5	Volatility State Variable	72
3.2.6	Practical Simulations	75
3.2.7	Compatibility with Non-Event Based Modelling	76
3.3	Model Analysis	78
3.3.1	Equivalence to Other Point Processes	78
3.3.2	Distribution after Delay	81
3.3.3	Calculation of the Equilibrium Point (Metastable Point)	83
3.3.4	Additional Discussion	84
3.4	RRAM Metal Oxide Device Example	86
3.4.1	Physical Model	86
3.4.2	Readout Equation	87
3.4.3	Dataset	88
3.4.4	Model Fitting	91
3.4.5	Simulation	96
3.5	Neuromorphic Applications	97
3.5.1	Synaptic Weight Decay through Resistive Drift	98
3.5.2	ReLU Nonlinearity for Memristive Neural Networks	99
3.5.3	Short and Long Term Memory	99
3.5.4	Spiking Behaviour through Volatility	102
3.6	Conclusions and future work	104
4	Conditional Generative Adversarial Networks for Modelling Resistive Drift in Mem-	
	ristors	107
4.1	Introduction	108
4.2	Resistive Drift Dataset	109
4.2.1	Parameter Selection and Dataset Creation	110
4.2.2	Dataset Visualisation and Evaluation	111
4.3	Conditional GAN Training	112
4.3.1	Generative Adversarial Network Training	113
4.3.2	Data Transformation and Normalisation	113
4.3.3	Feeding Data to the Network	116
4.3.4	Multiple Sample Discrimination	116
4.3.5	Delay Discriminator	117
4.3.6	Architecture	119
4.3.7	Training Parameters	119
4.4	Results	124
4.4.1	Delay Consistency Evaluation	124
4.4.2	Qualitative Evaluation	125

4.4.3	Conditioned Statistics Evaluation	127
4.5	Conclusion	128
5	Energy-Constrained Information Storage on Memristive Devices Over Time	131
5.1	Introduction	131
5.1.1	Information Storage as a Communication Problem	132
5.2	Problem Formulation	133
5.2.1	Capacity of the Storage Channel	136
5.2.2	Energy-Retention Trade-off	136
5.3	Method	140
5.3.1	Data Normalisation	141
5.3.2	Regularisation	141
5.3.3	Conditional Generalised Divisive Normalisation	144
5.3.4	Additional Residual Delay Conditioning	144
5.3.5	Deep Joint Source-Channel Coder Architecture	146
5.3.6	Delay Specialisation	146
5.3.7	Training Parameters	147
5.4	Results	147
5.4.1	Reconstruction Results	148
5.4.2	Single Delay and Noiseless Ablation Experiments	150
5.4.3	Energy Distribution	150
5.4.4	Ground Truth Model Evaluation	155
5.5	Conclusion and Future Work	155
6	Adversarial Jamming for Autoencoder Distribution Matching	157
6.1	Introduction	157
6.2	Problem Definition	159
6.2.1	Jamming as Regularisation for an Autoencoder	162
6.3	Method	163
6.3.1	Data Autoencoder Architecture	163
6.3.2	Shared Randomness	165
6.3.3	JSCC Residual Architecture	165
6.3.4	Normalisation Constraints	165
6.3.5	Metrics	167
6.3.6	Ablation Architecture and Regularisation	168
6.3.7	Datasets and Optimisation Parameters	169
6.4	Results	169
6.5	Discussion	169
6.6	Conclusion	174

7	Conclusions and Future Work	175
7.1	Summary of Contributions	175
7.2	Limitations and Future Work	176
7.3	Concluding Remarks	178
	Bibliography	181
A	Multiple Sample Discrimination Ablation Results	195

LIST OF FIGURES

Figure 1.1	Humans have sought to simplify their lives since prehistoric times through the use of tools. The transition from automation of physical processes to the automation of mental processes heralded a new era of tool use and tool making, setting the stage for what was to evolve into the general field of Artificial Intelligence (AI), and particularly the opportunities afforded by machine learning and - specifically - deep learning. From left to right: stone age tools, an abacus, Babbage's difference engine, and ChatGPT.	26
Figure 2.1	The computational primitives of neurons enable them to perform intelligent computation in an energy-efficient manner.	36
Figure 2.2	A simplified diagram showing the typical structure of a memristive device: a thin layer of semi-conducting or partially insulating material sandwiched between two electrodes. The conductivity of the active layer can be modulated through the use of an applied bias or current through a variety of physical, chemical, or electrical changes.	41
Figure 2.3	An illustrative example of a rate-distortion function for a source with a particular distortion measure, showing the region of achievable rates for given distortions and the entropy of the source as the rate where any given distortion for a source will always be 0. $R(D)$ is the rate-distortion function, yielding the minimum achievable rate R , as a function of an upper bound, D , on the distortion. $H(X)$ is the entropy of the source distribution for the random variable X	51
Figure 2.4	A figure demonstrating the roles of source and channel coding. Source coding compresses information, while channel coding introduces redundancy to code against errors. In joint source channel coding, increased compression can be traded off for increased redundancy or vice versa, while maintaining a constant rate of transmission and the source and channel encoders are combined into a single encoder (as are the associated decoders).	52
Figure 2.5	Generative modelling techniques fall into a range of categories, seen in this diagram.	57
Figure 3.1	A model's use is governed by its predictive ability in a particular setting. We focus on memristor modelling appropriate for use in a neuromorphic computing setting, introducing event-based simulation and the use of a volatility state variable to enable the model to be used to simulate key behaviours necessary for this.	62

Figure 3.2	The metastable switch model. The conductance is a linear function of the number of switches (parallel conducting channels) that are in the “on” state at any one time.	64
Figure 3.3	A diagram showing an example of the state transitions of a single switch for $k = 4$ states. The readout equation $R(\mathbf{r})$ is a function of the state vector denoting the instantaneous states of all of the metastable switches $s_i, i \in [1 \dots N]$	66
Figure 3.4	According to Chua’s original formulation, there should be a one-to-one relation between either the charge or the flux and the memristor resistance (memristance). For memristive systems, there may be multiple state variables, not necessarily including charge or flux, that determine the memristance, and the state vector and memristance are in a many-to-one relationship (that is, the states cannot necessarily be fully characterised from the memristance alone).	68
Figure 3.5	Illustration of the effect of the different parameters on the Boltzmann barrier heights in the forward and reverse switching directions. In a) we see the standard barrier, unmodulated by an offset or an applied voltage. b) shows the barrier modified by an energy offset E_{off} . c) shows the energy barrier modulated by the offset in addition to the modulation of the barrier resulting from the application of a voltage bias, E_v . The relative energy barrier height for each local minimum determines the respective Poisson switching rates λ'_{up} and λ'_{down}	71
Figure 3.6	Our physical model for switching in a titanium dioxide memristor. The growth of the filament will not significantly impact the resistance (conductance) - i.e. the readout value - until the filament size crosses a particular threshold. Once this condition is achieved (middle figure), we are in the active region of switching, where small changes in the state variable have a significant impact on the resistance, due to the modulation of an effective area of conduction through the active layer of the device.	87
Figure 3.7	The active switching region is defined and shown as the region where changes to the state variable measurably impact the readout variable (in this case the resistance). Before this point, changes to the state are modelled not to have an impact.	88
Figure 3.8	The unquantised resistance dataset, showing the empirical distribution of all resistance values in all series.	90
Figure 3.9	Figures showing the statistics of the generated pairs dataset.	91
Figure 3.10	The mean change in resistance for the dataset of pairs with $\Delta T = 10000$. We show a moving average of the data with different averaging window sizes (5 and 50), which give a clearer picture of the nonlinear trend. The linear conductance model with best fit parameters is plotted alongside the data and trend lines.	93

Figure 3.11	Fitting the event-based model to the linear conductance model. We use a constant rate assumption to derive approximate values for the parameters.	94
Figure 3.12	The mean change in the resistance following fitting to the linear conductance model, with errorbars indicating the standard deviation of series, for simulations with length $\Delta T = 10000$	95
Figure 3.13	Simulated experiments for a variety of different stimulating voltage inputs. From the alternating pulses experiment, it is clear that the preferred direction of movement (the equilibrium resistance) being a high value, results in the final resistance being higher than the initial resistance, due to the impact of resistive drift, despite the same number and magnitude of positive and negative pulses being applied.	98
Figure 3.14	Frequency experiments for a starting state of $20k\Omega$: we see that despite the same total amount of energy being applied to the devices (the same magnitude voltage applied over the same total amount of time), we achieve a higher metastable resistive state when using a higher frequency input, corroborating frequency-dependent switching results observed in the literature. Note that this is a result of the larger magnitude attained by the volatility state variable, $\rho(t)$, for higher frequency inputs, which accelerates switching for a given bias.	101
Figure 3.15	The impact of $\rho(t)$ on n_{eq} , for different bias values. Parameters for the calculation are drawn from Table 3.3. As $\rho(t)$ increases, the equilibrium point becomes closer to $N/2$	103
Figure 3.16	Spiking current behaviour produced using model parameters that produce oscillatory behaviour, made possible through the volatility state variable. Spiking behaviour can be produced through appropriate parameter selection. The intensity of the spikes, the number of spikes in the train, and the frequency of the resultant spike train are dependent on the magnitude of the applied input.	104
Figure 4.1	An illustrative subset of the series in the dataset used for training the conditional Generative Adversarial Network (cGAN), generated using the event-based model of Chapter 3, used to illustrate the stochasticity and the convergence to the equilibrium point of $500k\Omega$. Note the increase in stochastic nature of the drift as a function of the magnitude of the resistance.	111
Figure 4.2	Histogram of final resistance values in the drift dataset at different delays. We see that as the delay increases to 1000s, the distribution becomes centred around the equilibrium resistance value - i.e. any information present in the encoding of the initial value is lost. The legend shows the mean value for each delay conditioned distribution, μ , and the standard deviation, σ	112

Figure 4.3	Histograms of the values in the dataset, D following the initial pointwise application of the resistance normalisation transform (Figure 4.3a) and subsequent application of the differences transform for pairs of values from the dataset separated by a given delay (Figure 4.3b).	115
Figure 4.4	Structure of the cGAN used for training. The delay processor and resistance processor map the delay d and the normalised resistance at time t , denoted by $\bar{r}_{init}$, to independent embeddings. The combined processor network then processes these embeddings along with the diagonal Gaussian noise vector of dimension z , which is the latent prior for the generative model. The output of the combined processor is a normalised difference, to which the inverse differences transform is applied, followed by addition to $\bar{r}_{init}$ to produce the predicted resistance after delay d , $\bar{r}_{final}$	120
Figure 4.5	Structure of the discriminator architecture and the multiple sample discrimination approach. The condition processor processes the first element of the sequence, which is simply the initial condition, in addition to the delay. The remaining elements of the sequence vector: $r_2 \dots r_s$, are passed to the sequence processor, along with the output of the differences normalisation transform applied to each sequence. The output embeddings of both processors are concatenated and passed to the combined processor. When using the technique of multiple sample discrimination, we pass n sequences (and n delays) to the network simultaneously, with the condition processor and sequence processor input dimensions being scaled up by n and the effective batch size for the discriminator being scaled down by n . The same architecture is used for the main discriminator and the auxiliary delay discriminator described in Section 4.3.5.	121
Figure 4.6	The training structure for the generator and the two discriminators. This visualises the delay discrimination procedure. For the purposes of clarity in visualisation, we do not show multiple sample discrimination in this figure. The top of the diagram shows the generation of a fake sequence for the canonical adversarial training, with the discriminator trained to distinguish between real and fake resistance sequences. The bottom of the diagram shows the training of the cGAN to enforce consistency in its output for the same total delay, also addressing the problems with closed loop training described in Section 4.3.5.	122
Figure 4.7	Experiments examining the consistency of the model for generation of outputs of the same cumulative delay, ΔT , at different levels of delay refinement d . We measure the mean change in the resistance for a variety of starting resistance values. In the case of a delay of d , we use $500/d$ steps run in order to achieve a total equivalent delay of $\Delta T = 500$	125

Figure 4.8	Generated series for different delays with (left figures 4.8a, 4.8c, 4.8e, 4.8g) and without (right figures 4.8b, 4.8d, 4.8f, 4.8h) the delay discriminator, for an equivalent delay of 1000, recurrently evaluated for different delay conditions $d \in \{1, 10, 100, 500\}$	126
Figure 4.9	Means and standard deviations of the final resistance conditioned on different combinations of initial resistances and delays, for the generator trained using the delay discriminator (Figure 4.9a and 4.9b) and trained without the delay discriminator as an ablation experiment (Figures 4.9c and 4.9d). The empirical statistics for each condition are plotted alongside those obtained from the ground truth model, in order to demonstrate the ability of the model to fit to the true distribution. We see a marked improvement in the statistical moment matching when using the delay discriminator. Legend entries denote the delay, d that the statistics are evaluated for, with ‘real’ entries being the statistics derived from the ground truth event-based model.	127
Figure 4.10	Empirically sampled final resistances for given initial and final resistance conditions. Figure 4.10a shows the statistics for the ground truth event-based model. Figure 4.10b shows statistics for the cGAN. Figure 4.10c shows statistics for the cGAN model evaluated recurrently with a delay of $d/10$ and 10 recurrent steps. Figure 4.10d shows an ablation experiment run without the delay discriminator, and Figure 4.10e shows a recurrent evaluation of the delay discriminator ablation experiment with a delay of $d/10$ and 10 recurrent steps.	129
Figure 5.1	Communication of information occurs through both space and time. In traditional wireless communication, the majority of the challenge is associated with transmission and recovery of information through space, whereas in the case of storage, we are more concerned with how the passage of time affects the quality of the recovered information. In both cases, distortions are introduced by the conditional channel distribution, for a given source alphabet.	132
Figure 5.2	Costs associated with bringing the resistance of a memristor to the specified value, starting from $R_{\text{init}} = 500\text{k}\Omega$, $\tau_{\text{final}} = 1.0$, $K = 10.0$, and $R_{\text{final}} = 1\text{k}\Omega$. We see that the energy costs marginally increase as the desired final resistance is lowered, whereas they marginally increase as the desired final resistance is increased beyond R_{init} . The right hand plot shows the trade-off plotted in terms of the resistance, whereas the left hand plot shows the trade-off in terms of the conductance, on a logarithmic scale, where the trade-off is linear, demonstrating the exponentially increasing marginal costs of bringing the device into a lower resistive state.	140
Figure 5.3	Histogram of the delays following the logarithmic transformation used for training and conditioning the autoencoder.	142

Figure 5.4	The Conditional Gaussian Divisive Normalisation (CGDN) transform for conditioning the encoder and/or decoder network on the delay, applied to the latent channels outputs for each convolutional layer for the encoder/decoder conditioned settings.	145
Figure 5.5	The Deep Joint Source-Channel Coding (Deep JSCC) autoencoder architecture used for training, based on modifying an existing architecture to the delay-conditioned setting Bourtsoulatze et al. (2019). In our model, we replace the Generalised Divisive Normalisation (GDN) layers with CGDN layers, as well as introducing fully connected residual delay processors at the output of the encoder and/or the input of the decoder, depending on the delay conditioning setting. CGDN and the delay processors are only included for the encoder/decoder, respectively, if the setting includes such delay conditioning. For unconditioned settings, we make use of GDN transforms and do not include the fully connected delay processors. In the diagram c_i are the number of colour channels in the image, and c_l are the number of latent channels. Changing c_l modifies the compression rate of the autoencoder.	146
Figure 5.6	A comparison of the reconstruction errors for a variety of autoencoder experiments, demonstrating the impact of delay conditioning of the encoder/decoder networks. Introducing delay conditioning to the encoder, allows the autoencoder to allocate different energy budgets to different conditions, while still respecting the average energy budget. Decoder conditioning also results in improved results compared to the absence of conditioning, with the decoder able to correct for delay-dependent distortions, resulting in better SNR reconstruction values across a wider range of the delays considered.	149
Figure 5.7	Image reconstruction results for a range of delays, for the different delay conditioned settings, for an energy budget of 0.1J.	151
Figure 5.8	Image reconstruction results for a range of delays, for the different delay conditioned settings, for an energy budget of 0.01J.	152
Figure 5.9	Comparison of the encoder/decoder/encoder and decoder -conditioned settings for autoencoder training to a range of ablation experiments, for training, without delay conditioning, on single delays only (1, 10, 100, 1000) and on a noiseless channel. We see that although training for a particular channel setting (delay) can improve the performance for that particular condition, the performance is much worse across other delays. The legend displays the average energy consumption for each of the scenarios.	153

Figure 5.10	A comparison of the input distributions for the different encoder/-decoder conditioned settings, for two different energy budgets: 0.1J, and 0.01J. We see that in the unconditioned cases, and the decoder conditioned cases, the histograms are the same across delays, whereas encoder and decoder conditioning leads to adaptation of the input distribution, with input distributions that are more costly in terms of the energy budget being allocated to larger, noisier, delay conditions. Note that the counts are shown on a logarithmic scale for improved clarity.	154
Figure 5.11	Evaluation on the ground truth event-based model for the reconstruction Peak Signal to Noise Ratio (PSNR) of the autoencoders trained for an energy budget of 0.01J on the cGAN model from Chapter 4. We also show the energy consumption of the input distribution for each delay and input resistance condition. We plot the performance when evaluated on the cGAN model alongside these results for comparison. We see that for some delays, there is drop in performance when we switch from evaluation on the trained cGAN model to evaluation on the ground truth event-based model, highlighting regions where the statistics were not matched perfectly. However, the overall match in the level of performance is seen to be good.	156
Figure 6.1	Adversarial Jamming (AJ) for prior distribution matching. The game between a compressor/jammer f (compressing the images) and a Joint Source-Channel Coding (JSCC) autoencoder (g, h) (for source $\mathbf{X} \sim \mathcal{N}(\mathbf{o}, \mathbf{I})$) has a saddle point with $\mathbf{Z}^* \sim \mathcal{N}(\mathbf{o}, \mathbf{I})$, implicitly imposing this prior on the latent space.	158
Figure 6.2	AJ can be seen as an alternative regularisation term in the framework of distribution matching regularisation for generative autoencoders. We derive the worst case additive jamming noise for an adversarial game between a transmitter/receiver pair and an adversarial jammer, based on theoretical properties of the noise that are given in Akyol and Rose (2013). Using the learned representations as jammer noise, we enforce a game whose equilibrium is such that the representations match the required Gaussian prior distribution.	160
Figure 6.3	Framework used for regularisation using the AJ problem. The specific architecture for the data autoencoder (compressor and reconstructor) is detailed in Figures 6.4. We make use of a residual and symmetric fully connected autoencoder architecture with residual connections for the Deep JSCC autoencoder (transmitter and receiver) as detailed in Section 6.3.3.	163

Figure 6.4	Architecture of the Deep Convolutional Generative Adversarial Network (DCGAN)-style convolutional compressor (data autoencoder's encoder) and reconstructor (data autoencoder's decoder). In the figure, c_i is the number of colour channels in the image, while c is a parameter to be chosen, dictating the number of latent channels. n dictates the number of convolutional operations to be performed. The convolutional and deconvolutional layers all have the same filter size. Note that each of the outputs of convolutional or deconvolutional layers is batch normalised and then passed through a pointwise ReLU activation function following the convolution or deconvolution operation.	164
Figure 6.5	Deep JSCC architecture for the transmitter and receiver. We use residual, or skip, connections in order to enable the easier flow of information and gradients, allowing for the network to parametrise both a probabilistic linear mapping, which is the theoretical saddle point of the minimax communication game, as well as more complex transformations. . . .	166
Figure 6.6	Generation results across the three datasets (top to bottom: CelebA, CIFAR-10, MNIST), for AJ, as well as ablation experiments which replace the proposed AJ regularisation with a Wasserstein Autoencoder (WAE) Maximum Mean Discrepancy (MMD) regularisation loss, or a Variational Autoencoder (VAE) acKL Divergence regularisation loss. The dimension of the latent space is 8 for the MNIST results, and 64 for both the CelebA and CIFAR-10 results.	170
Figure 6.7	Reconstruction results across the three datasets (top to bottom: CelebA, CIFAR-10, MNIST), for AJ, as well as WAE MMD and VAE ablation experiments.	171
Figure 6.8	Interpolation results for the autoencoder trained using AJ as the distribution matching regularisation term for the CelebA dataset. The interpolation results demonstrate the coverage of the latent space between encoded representations. We see that the data autoencoder is able to smoothly interpolate between images in the latent space through linear interpolation of the compressed representations and corresponding reconstruction.	172
Figure A.1	Means and standard deviations of the final resistance conditioned on different combinations of initial resistances and delays, for the value of $n = 1$ for multiple sample discrimination for both the main discriminator and the delay discriminator. These can be compared to those in Figure 4.9 of Chapter 4, highlighting the improved moment matching when using a value of $n > 1$ for multiple sample discrimination. . . .	195

Figure A.2	Empirically sampled final resistances for given initial and final resistance conditions. Figure A.2a shows the statistics for the ground truth event-based model. Figure A.2b shows statistics for the cGAN. Figure A.2c shows statistics for the cGAN model evaluated recursively with a delay of $d/10$ and 10 recursive steps. These results can be compared to those in Figure 4.10 of Chapter 4.	196
------------	---	-----

LIST OF TABLES

Table 3.1	Drift dataset individual experiments. Each experiment involved one of the two arrays available. The memristors on the array were randomly programmed to values within the operating range, then periodically monitored for a number of hours (with one resistance reading taken every 10 seconds).	89
Table 3.2	Training and test errors for fitting the proposed linear conductance model, the linear resistance model, and the stretched exponential model to the pairs dataset, for two different dataset splits.	92
Table 3.3	Table of parameters for the model, for titanium dioxide memristive devices.	96
Table 3.4	Parameters for the input signals in the simulated switching experiments. In the case of the positive descending experiment, we observe that too large a signal amplitude drives the memristor immediately to a high resistance state, so we lower the initial amplitude to better observe the behaviour. In the case of the alternating experiment, we use a small duty cycle such that the active pulse duration is 1s.	97
Table 3.5	Parameter updates to produce spiking behaviour as shown in Figure 3.16, as compared to those in Table 3.3.	103
Table 4.1	The parameters used for generation of the simulated resistance drift dataset, with an equilibrium resistance of 500k Ω	110
Table 6.1	Architectural parameters for the data autoencoder, for each dataset trained on.	164
Table 6.2	Fréchet Inception Distance (FID) scores, Mean Squared Error (MSE) losses for image reconstruction, and the Determinant of the Pearson Correlation Matrix (DPC) scores. In the case of the FID and the MSE, lower is better, whereas for the DPC, higher is better (maximum 1.0). These are shown for the three datasets. In each case, we show results for AJ, a standard VAE, and a WAEs, trained using the same parameters and architecture. We see that the performance of AJ is comparable to WAEs and outperforms VAEs in every case. k denotes the latent dimension (representation size). Boldface entries denote the best performance.	173

LIST OF ALGORITHMS

Figure 1	Procedure for simulating the switching events of the event-based model.	76
Figure 2	Event-based modelling procedure, incorporating additional non-event-based inputs.	79
Figure 3	Procedure for conversion of the non-uniform time series to uniformly sampled time series.	80
Figure 4	Algorithm for updating the network parameters of the generator and delay discriminator for a single batch of size b . α represents the learning rate of the optimisation strategy. Note that we show the algorithm here for a value of $m = 1$ according to multiple sample discrimination, in order to simplify the notation and improve clarity. In practice, we use a value larger than 1.	118

ACRONYMS

AWGN Additive White Gaussian Noise
AJ Adversarial Jamming
AI Artificial Intelligence
BN Batch Normalisation
BCE Binary Cross-Entropy
CBN Conditional Batch Normalisation
CGDN Conditional Gaussian Divisive Normalisation
cGAN conditional Generative Adversarial Network
CPU Central Processing Unit
DCGAN Deep Convolutional Generative Adversarial Network
Deep JSCC Deep Joint Source-Channel Coding
DPC Determinant of the Pearson Correlation Matrix
ECM Electrochemical Metallization
EEG Electroencephalography
ELBO Evidence Lower Bound
FID Fréchet Inception Distance
fMRI Functional Magnetic Resonance Imaging
GAN Generative Adversarial Network
GDN Generalised Divisive Normalisation
GMSM Generalised Metastable Switch Model
GPU Graphics Processing Unit
GRU Gated Recurrent Unit
GST Germanium Antimony Tellurium
IMQ inverse multi-quadratics
JSCC Joint Source-Channel Coding
KL Divergence Kullback-Leibler Divergence
LSTM Long Short-Term Memory
LTP Long Term Potentiation
MIM Metal-Insulator-Metal
MMD Maximum Mean Discrepancy

MSE Mean Squared Error
PCM Phase Change Memory
PSNR Peak Signal to Noise Ratio
RRAM Resistive Random Access Memory
SNR Signal to Noise Ratio
STDP Spike Timing Dependent Plasticity
STP Short Term Potentiation
STT Spin Transfer Torque
SGD Stochastic Gradient Descent
TCM Thermochemical Switching Mechanism
TPU Tensor Processing Unit
VAE Variational Autoencoder
VCM Valence Change Mechanism
VQA Visual Question Answering
WAE Wasserstein Autoencoder

INTRODUCTION

Since the beginning of conscious existence, humans have been searching for ways to simplify their lives. We have created and made use of tools to make it easier for us to carry out tasks involving manual labour. The use of tools is one of the defining characteristics of being human; tools enable us to perform the tasks necessary in our daily lives with a greater degree of ease. From the simple notion of using objects in the environment to help us carve stones, to building fires to cook our food and keep us warm, human beings have been exploiting the physical world that they exist in for millennia in order to automate routine tasks and to enable what would be otherwise impossible to achieve.

Tools exist not only in the physical world, but also in the mind. We, as human beings, have developed systems of thinking to frame our lives, in order to enable us to do material things that would not otherwise be possible. The term Artificial Intelligence (AI), in recent years, has become familiar and often overused (or even misused), and one could be forgiven for thinking that it is a relatively new development. However, AI - in its true sense - is a rather ancient technological movement, going back further even than the 20th century. It literally refers to the automation of our own thoughts (human intelligence) by making use of the external environment (artificial means). One key development that led to our ability to do this was the dawn of symbolic thinking: where one thing represents another. The physical beads on an abacus are used to represent the different digits of a number, and their movement along the rods constitutes computation using those numbers - automated thoughts. The creation of first mechanical (Babbage's difference engine), then digital computers enabled the vast acceleration of this principle of automation, through the representation of symbolic information as binary voltage levels in transistors. At first, even with the aid of digital computers, algorithms for the manipulation of symbolic information had - for the most part - to be manually programmed and designed; however, a fundamental shift came about with the invention of neural networks and backpropagation. These enabled the efficient optimisation of general purpose parameterised functions to perform specific mental tasks, simply by specifying the desired behaviour in terms of a differentiable loss function. We had begun to automate the process of automating (Figure 1.1).

Neural networks - and their training procedures - were historically developed and based on simplified models of the function of biological neurons, borrowing from them the key computational elements of nonlinear activation functions, credit assignment, and the strengthening and weakening of connections for learning. Since the creation of the first versions of these biologically inspired learning algorithms, there have been a range of engineering advances which have made them hugely powerful tools. Deep generative models in the fields of natural language processing and computer vision have demonstrated extraordinarily impressive results, fuelled by research breakthroughs such as transformer architectures Vaswani et al. (2017), generative diffusion models Ho et al. (2020); Song and Ermon (2020), and contrastive learning methods Chen et al. (2020). Yet despite the rapid growth and success of modern intelligent

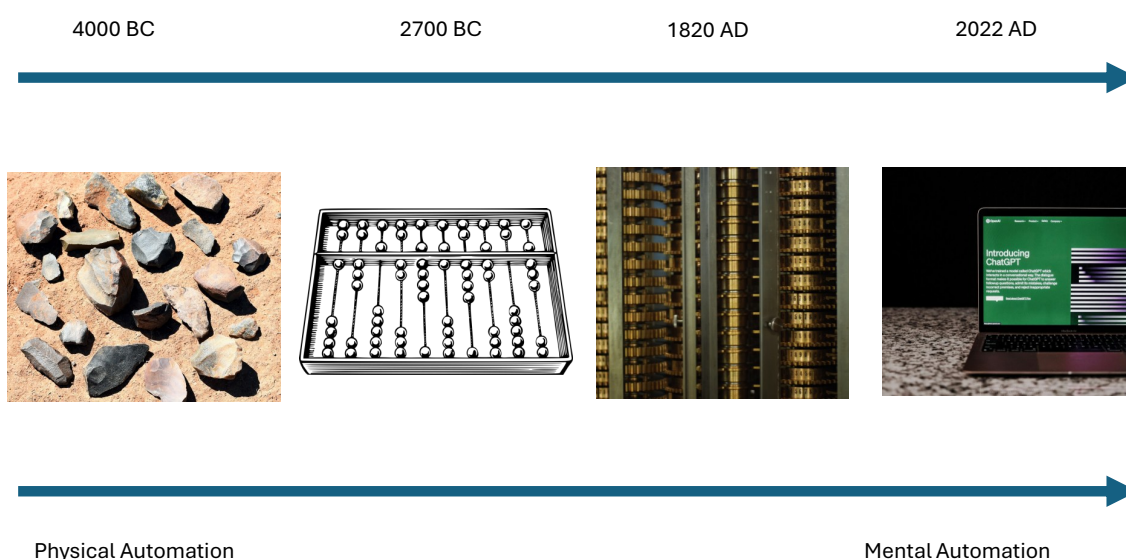


Figure 1.1: Humans have sought to simplify their lives since prehistoric times through the use of tools. The transition from automation of physical processes to the automation of mental processes heralded a new era of tool use and tool making, setting the stage for what was to evolve into the general field of [AI](#), and particularly the opportunities afforded by machine learning and - specifically - deep learning. From left to right: stone age tools, an abacus, Babbage's difference engine, and ChatGPT.

algorithms in machine learning, the fundamental tenets that support the field have remained relatively constant: firstly, the the use of neural networks as general, learnable, parameterised functions; and secondly, the use of gradient descent to update the neural network parameters to achieve a desired learning goal.

1.1 WHAT IS INTELLIGENCE?

The question of determining how, and to what extent, it is possible for us to build machines that think for us is a proxy for another question - what exactly the words “thinking” and “intelligence” constitute. Thus, the process of building computers to automate mental tasks has been underpinned by better understanding which components of our own brains are necessary to enable these mental tasks, and which are just biological fluff.

“*What is Intelligence?*” is a question that may seem to border on the philosophical. Ultimately, however, intelligence is a concept which has been artificially defined by human beings. We intuitively understand what intelligence is because it is a collection of phenomena which we ourselves have decided to label as such. In aiming to formalise a definition of intelligence, the same notion has been expressed in several different contexts. It has been described as information compression or entropy minimisation [Friston \(2010\)](#); [Friston et al. \(2006\)](#); [Schrödinger \(1992\)](#), the ability to predict the future [Marino \(2022\)](#), or simply the ability to deal with the environment in order to survive. Intelligence is our ability to model the environment

in a way which is both accurate and *efficient*. It is the ability to produce predictions that are useful for the survival of the predictor, and - and this is key - to do this in an efficient manner in terms of the time and resources (energy) used.

1.2 ACCURACY AND ENERGY

Everything is a model. Human beings interact with the world around them through inventing models of the environment - often causal - and acting on them. The nature of a model can be understood by understanding the reason behind the model's creation. Models exist in order to simplify the world around us, and act to allow us to predict future behaviour from past behaviour in an **efficient** and **expedient** manner. The nature of the model reflects the purpose of the predictions. For example, we can think of a dog whose objective is to receive a treat from a human being. Their model of the world encompasses the way in which the environment around them responds to their actions. For example, they might find that looking up at their human owner in a certain way tends to relatively reliably elicit a certain sequence of events that leads to their getting fed. The model does not necessarily explain why this happens, but it achieves its purpose in getting the treat.

For a human being who wishes to invent drugs that interact with the human body, the model of the environment they develop might consider the interactions between molecules, cells, and chemicals at a biological level of abstraction. Although their model cannot exactly predict the environment, it can predict what is necessary to achieve the goal - namely, improving the health of humans and extending their lifespans. In contrast, a model applied to the same phenomena at a subatomic, or even quantum, level of abstraction may be able to ultimately much more accurately predict the interactions, however the actual simulation of the model may be hugely inefficient and expensive relative to the marginal benefits accrued in terms of the attainment of the objective of designing good medicines. This highlights the need for models to take into account not just prediction accuracy, but also the energy and time required to make the predictions.

Although the results achieved by modern intelligent algorithms have been impressive, they have been extremely expensive to attain. Speed-ups and efficiency increases are provided by using more specialised hardware tuned to the matrix multiplication operations on which most deep neural networks are built, using Graphics Processing Units (**GPUs**) or Tensor Processing Units (**TPUs**) instead of less specialised Central Processing Units (**CPUs**). Despite this, the training process still often requires eye-watering sums of investment to support the level of energy consumption and time necessary. This brings into question the sustainability of the trajectory of modern intelligent algorithms based on running matrix operations on traditional hardware such as this.

Computer architectures as they exist today make sacrifices in terms of energy and computational efficiency, in order to allow for greater control, modularity, and certainty in computation. In digital logic, environmental nonlinearities and uncertainties are usually viewed as undesirable nuisances which should be got rid of. In contrast, analogue biological systems tend to operate in constantly changing and noisy environments. They are thus much more difficult

to control and design. Rather than simply attempting to correct for or align “noise” and complexity in the environment, biological information processors (i.e. neurons) bend their own processing rules and information to the system, exploiting the system as it exists and developing a processing paradigm based upon it, yielding huge rewards in energy efficiency, algorithmic possibilities, and information processing speed in the process, that together make it feasible for the brain to function.

Inspired by the efficiencies of the brain, we would like to be able to both exploit the physical processing capabilities and energy efficiency of carrying out computation in the natural world, and must be able to do this in a way which we can understand and control. One approach to computing which aims to use biological principles is neuromorphic computing. Underlying this is the principle of matching the algorithm to a potentially nonlinear and noisy system, rather than attempting to artificially transform the system to a noiseless state where an algorithm can run. We will explore whether in facing, rather than erasing, noise in the environment and making use of *computational primitives* as they exist in nature, we can design more efficient systems for intelligent computation in non-ideal environments.

1.3 COMPUTATIONAL PRIMITIVES

Although we can be forgiven for being lulled into thinking that using traditional binary and logic-gate based computing hardware is the *only* way to engineer and implement algorithms for intelligence, this is far from true. In fact, this is just one of many possible frameworks that could be used for computation. A wide range of elementary operations exist in nature, which we refer to as **computational primitives**. Each computational primitive can be thought of as a function in time or space, mapping an input to an output. If we consider the domain to be the electronic variables of voltage and current, then capacitors and inductors implement differentiation and integration, while resistors implement multiplication. More complex passive electronic devices may implement other operations, such as thresholding behaviour and exponential functions using diodes. Primitives can also exist at a system-level; for example, the system level behaviour of negative feedback, or that of competition, each of which can encourage a system to converge to a particular implicit stable point.

Approaches to computation based on alternative computational primitives have been traditionally prohibitive due to the complexity of designing principled methods for exploiting the wide variety of computational primitives available to us in their native form. As such, digital computing, which abstracts away the physical idiosyncrasies of a system, at the expense of overheads in time and energy, due to the mismatch between the calculus (the system for programming and algorithmic design) and the elementary operations (computational primitives) used to implement these algorithms. However, in the era of machine learning and deep learning, we are able to design algorithms which can easily specialise and adapt in a much more dynamic and data-driven way than was possible before.

In this thesis, we take a particular interest in the computational primitives that exist in the human brain. We may ask what are the core behaviours exhibited by biological neurons that enable them to so effectively bring about intelligent behaviour? In fact, a number of key

features of biological neurons, and systems of such neurons, that allow for fast and efficient intelligent computation can be identified. The principle of taking inspiration from the brain to implement computational algorithms is known as neuromorphic computing.

1.4 NEUROMORPHIC COMPUTING

The distillation of the fundamental elements of intelligent computation - the computational primitives - of the brain, while discarding any evolutionarily convenient biological “fluff” surrounding them, offers the potential for savings in terms of energy consumption and computational complexity. The process of biological evolution can be seen as a large scale random search across architectures, inductive biases, and parameters, leading to the honing of the human brain to its current form today. This is not to say that the human brain is perfect - humans are perfectly capable of making sub optimal decisions for their own survival - but the millions of years of tuning and re-tuning that have gone into the creation of the system as it stands should not be taken lightly.

Given that the brain has had so long to evolve its intelligent capabilities, and is itself an absolute reference for the definition of intelligence, it should be the first source of inspiration when we look to abstract intelligence away from the biological mess that exists within our heads - in other words, differentiating the essential components from the evolutionary conveniences and specific realisations of intelligent behaviour. In identifying the essential components, we coming closer to a *definition* of intelligence. Inspired by the capabilities of the human brain, the concept of neuromorphic computing was developed, involving building information processing devices inspired by the human brain, using its computational primitives.

In performing computation according to physical laws and removing as many artificial information processing barriers as possible, we achieve efficiency. A good practical illustration of this is the Pathfinder algorithm [Mead et al. \(1989\)](#), an algorithm for finding the shortest route between two destinations. The system made use of an analogue electronic circuit and negative feedback, to allow for a massively parallel distributed search using the laws of nature. Using the algorithm, an optimal path is found without any explicit optimisation, by virtue of stable point of the problem.

Taking inspiration from the human brain, a number of foundational features of neuromorphic computing [Mead \(1990\)](#) can be identified:

1. Event-driven processing of spatio-temporal data.
2. Computation using the laws of physics, avoiding inefficiencies associated with the artificial conversion from the digital to the analogue domain.
3. Massively parallelised processing of information: distributed computation.
4. Adaptation, for example through local averaging operations.
5. Fault tolerance and operation in noisy environments.

These neuromorphic principles can all be connected through the notion of the use of neural and neuronal computational primitives: using local functions or more abstract system-level

operations that exist in real devices and the natural world, in order to perform the computations that we desire, and to provide a basis for engineering new information processing algorithms. In doing so, we bend our algorithms to the existing computational primitives that exist, rather than attempting to transform the environment to enable it to run the computational primitives necessary for the algorithms.

Motivated by the effectiveness of the human brain in performing intelligent computation, in this thesis we will aim to investigate aspects of the foundational principles of biological computation using neural computational primitives, exploring how we might be able to use and exploit them for computation.

1.5 RESEARCH QUESTIONS AND OUTLINE

Though we could explore a wide range of ways in which intelligent computation could be accelerated and enhanced through adapting our algorithms to the computational primitives that exist in nature, the breadth of the search space makes this a hugely complex and challenging problem. As such, it is wise to focus on studying the importance and reproduction of the existing computational primitives of the human brain, which has already achieved a range of energy-efficient intelligent behaviours. We will look to identify several computational primitives of biological neurons, and systems of such neurons, that enable them to perform intelligent computation in an efficient and effective manner. We aim to show that the ideas of noise and nonlinearity form a basis for many these primitives, and as such endeavour to explore how we might be able to use and exploit them for computation and intelligent information processing. We thus formulate our research question for this thesis as follows:

“How can we store and process information in the presence of noise and neuronal nonlinearities?”

There exist two sides to this question: the first is the question of how we *adapt* to and cope with the noise and the nonlinear computational primitives that are present in a system, to prevent the degradation of performance. The second is how we *excel*, by not only learning to cope with, but indeed make use of these “non-idealities” to improve the performance of intelligent algorithms. In this thesis, we will explore this question from both these perspectives.

In Chapter 2, we will introduce the range of stochastic and nonlinear behaviours present in neurons and systems of neurons. In order to use and exploit such primitives, we detail artificial analogues for these nonlinearities, and methods for their implementation, in electronic devices called *memristors*. We will introduce communication and information theory as a framework for studying the problems of noisy computation and storage. To conclude the chapter, we will discuss the benefits of noise and adversarial noise for improving generalisation and robustness in learning algorithms.

In Chapter 3, we begin our investigation of the research question, by studying the conditions for implementation of a number of neuronal computational primitives using memristors. We demonstrate, through a neuromorphic modelling approach, that nonlinear bias and temperature dependent drift dynamics, together with a nonlinear phenomenon known as the volatility,

enable a number of behaviours including synaptic weight decay, frequency-dependent potentiation, and action potential-like spike generation with a response that is proportional to the input signal magnitude.

Motivated to investigate the limitations of storage and processing in the presence of the different nonidealities of memristive devices, in Chapter 4, we focus on modelling the phenomenon of resistive drift, as an analogue for synaptic weight decay in neurons. We look at how generative modelling techniques can allow for efficient and differentiable modelling. As such, we introduce a data-driven deep learning-based generative model and explore techniques for modelling the delay-conditioned distribution of the decay of information content over time.

In Chapter 5, we delve further into the phenomenon of resistive drift. We explore how stochastic resistive drift in memristive devices can be quantified through a fundamental trade-off between the energy expended in storing information on the devices and the stability of the information - i.e. the capacity of the memristor as a communication channel. We investigate an approach to building autoencoders to trade-off the energy consumption for the information content, with awareness of the delay to recovery of the information, demonstrating that incorporating knowledge of the delay can enable more optimal use of the device capacity under energy constraints.

Finally, in Chapter 6, we look at noise in neural networks, in the context of a representation learning problem for generative models. We build upon previous insights into the benefits of noise for generalisation and regularisation, and explore the use of the phenomenon of the worst-case type of noise: adversarial noise. We demonstrate that such noise can be used to impose a maximally informative diagonal Gaussian distribution on the learned representations of a generative model, unveiling a new perspective on how noise can not only be dealt with, but also exploited as a computational primitive, for generative modelling in the natural world.

1.6 CONTRIBUTIONS

In Chapter 3, we introduce two major contributions to memristive modelling for neuromorphic applications. Our first contribution is to look at a new simulation paradigm for event-based systems which is natively event-driven, based on the evaluation of time-to-fire for multiple parallel Poisson point processes, while also allowing for continuous-time changes - such as changes in temperature or bias - to be incorporated. We extend an existing generic memristor model, the Generalised Metastable Switch Model (GMSM) Molter and Nugent (2016); Nugent and Molter (2014) to the event-based setting, in order to address problems associated with time-stepped memristor switching simulations, where error is inevitable and proportional to the size of the time-step. We modify the rate calculations of the model, to base them on a more physically generic Boltzmann probability density function and also modularise the modelling strategy through the separation of the resistance and the state variables via a readout equation. Our second modelling contribution is to generalise a variety of metastable and autonomous evolution phenomena observed in different types of memristive device - thin film Metal-Insulator-Metal (MIM) devices and Phase Change Memory (PCM) - and propose the use of a single variable called the “volatility state variable” to singularly describe all of

these phenomena. Following the development of the model, we introduce a real resistive drift dataset for a titanium oxide memristive device, and demonstrate the ability of the model to reproduce the phenomenon of resistive drift. In the final section of the chapter, we provide demonstrations of ways in which the proposed modelling framework can be used for neuromorphic simulations, quantifying the ways in which biological phenomena - namely, frequency-dependent switching and the emulation of action potentials - can be brought about, through analysis of model behaviour and the appropriate selection of model parameters.

Following the development of the event-based model, in Chapter 4, we introduce a number of architectural contributions to enable the use of Conditional Generative Adversarial Networks (GANs) for the problem of modelling resistive drift. We highlight effective data pre-processing methods for enabling effective learning on data belonging to large and unbounded positive ranges, such as resistance values and delays. We introduce a continuously conditioned GAN for the generation of memoryless (resistive drift) time series that is conditioned on the delay, introducing a novel technique that we call “delay discrimination” which uses an auxiliary adversarial network to enforce consistency in the model outputs for different timescales, and to enable generalisation outside of the range of delays present in the training data.

In Chapter 5, following the development of a differentiable generative model for resistance drift in the memristors, we introduce a new, fundamental trade-off between the amount of energy consumed and the capacity of the devices for information storage. We then look at the use of this trade-off for storing information on memristive devices, exploring a novel technique for conditioning an autoencoder to enable adaptation to continuous channel conditions (the delay) that we call Conditional Gaussian Divisive Normalisation (CGDN).

Finally, in Chapter 6, we examine the exploitation of adversarial noise for intelligent computation in the context of generative modelling. We propose a new adversarial jamming approach to distribution matching based on a minimax game between a jammer and an auxiliary Joint Source-Channel Coding (JSCC) encoder and decoder pair to impose a diagonal Gaussian distribution on the latent space of an autoencoder. We show that this technique is effective for distribution matching in the latent space, comparable to generative models of the same kind: namely, Variational Autoencoders (VAEs) Kingma and Welling (2014) and Wasserstein Autoencoders (WAEs) Tolstikhin et al. (2018). To the best of our knowledge, this is a novel result that is the first of its kind to exploit a game involving a communication theoretic saddle point to regularise the latent space of an autoencoder for distribution matching, offering a broadened horizon for future algorithms that make use of noise and adversarial noise as computational primitives.

In the next chapter, we will provide an overview and background of several topics that will serve as a foundation for our contributions.

1.7 PUBLICATIONS

The content of this thesis is based on the work presented in several publications, listed below:

El-Geresy, W., Papavassiliou, C. and Gündüz, D. (2024). Event-Based Simulation of Stochastic Memristive Devices for Neuromorphic Computing. *arXiv preprint arXiv:2407.04718*.

El-Geresy, W., Papavassiliou, C. and Gündüz, D. (2024). Delay Conditioned Generative Modelling of Resistive Drift in Memristors. *arXiv preprint arXiv:2408.01539*.

El-Geresy, W., Papavassiliou, C. and Gündüz, D. (2024). Energy-Constrained Information Storage on Memristive Devices in the Presence of Resistive Drift. *arXiv preprint arXiv:2501.10376*.

El-Geresy, W. and Gündüz, D. (2024). Adversarial Jamming for Autoencoder Distribution Matching. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) 2024*

In this chapter, we aim to provide background for the approaches developed in this thesis. We begin, in Section 2.1, by introducing the computational primitives (fundamental operations used for processing) of neurons and the mechanisms by which they occur, to better understand how noise and nonlinear computational primitives are used by neurons in the brain for information processing and storage, as well as why they are essential features of intelligent computation.

In Section 2.2, we turn our attention to a class of passive electronic device called the memristor, which has been proposed as a possible neuromorphic computing component, replicating certain aspects of the behaviour of neurons. We detail the types of devices available, as well as their switching and conduction mechanisms, as background. We investigate existing approaches to the implementation of neural computational primitives using memristors in Section 2.2.4.

Since our focus is on the storage and processing of information in the presence of noise and nonlinearity, we look at prior approaches to the modelling of memristive devices for these purposes in Section 2.3, motivating the development of new models for the purposes of neuromorphic computing and for use in end-to-end differentiable deep learning models for storage and computation.

The study of noise and nonlinearity in such devices and beyond would not be possible without the theoretical tools of information theory and communication theory, specifically, the notions of source and channel coding, for which an overview will be given in Section 2.4. As part of this section, we look at communication theoretic and deep learning -based approaches to the problem of the storage of information on memristive devices in the presence of different kinds of noise in Section 2.4.2. We then introduce the notions of communication games and worst case noise in Section 2.4.3.

In Section 2.5, we introduce the notion of generative modelling, which is used in Chapter 4 to aid in the creation of a differentiable model of drift in memristors. This will also provide background and context for Chapter 6, where we attempt to improve upon existing generative modelling techniques.

Finally, in Section 2.6, we look at the general benefits of noise in deep learning and Artificial Intelligence (AI) for robustness and generalisation, further motivating the approach taken in Chapter 6, where we will introduce a novel technique that exploits the use of adversarial noise for the distribution regularisation of the latent space of autoencoders in generative modelling problems.

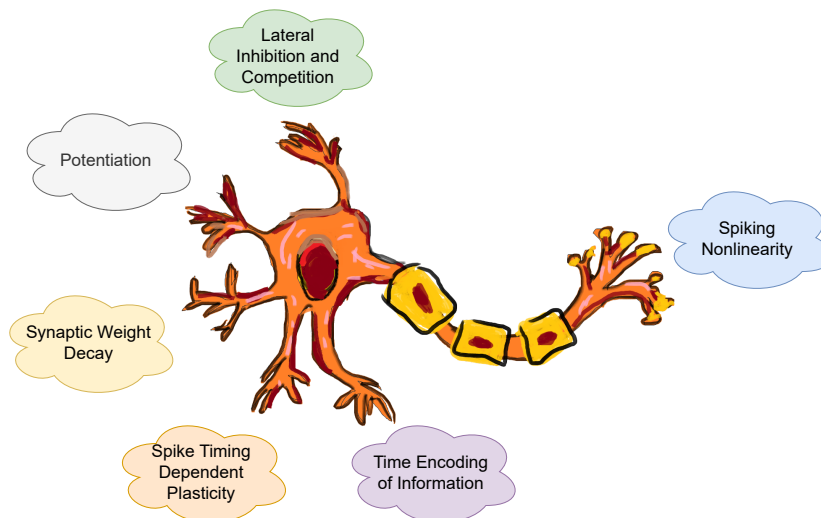


Figure 2.1: The computational primitives of neurons enable them to perform intelligent computation in an energy-efficient manner.

2.1 NEURONS AND THEIR COMPUTATIONAL PRIMITIVES

A range of nonlinear dynamics are present in neurons (Figure 2.1). Neurons operate in noisy environments, leading to the need for schemes for information storage and processing that are robust to the noise present in the environment. Early work involved understanding and modelling the operation of neurons, including experiments conducted by Hodgkin & Huxley [Hodgkin and Huxley \(1952\)](#) which resulted in the development of early electrical models of neurons and the unveiling of experimental evidence for action potentials as a fundamental component of information transmission. In addition to electrical signals, chemical neurotransmitters were also later found to play an important role in neuronal communication. Since then, techniques such as Electroencephalography (EEG) and Functional Magnetic Resonance Imaging (fMRI) have allowed for the real time monitoring of the activity of neurons and networks of neurons in the brain with higher degrees of spatial and temporal precision, resulting in the painting of a much more detailed picture of their dynamics [Purves \(2018\)](#).

The Hodgkin-Huxley neuron model attempted to accurately model the generation of spiking behaviour using a circuit involving differential equations to describe the movement of ions across voltage-gated ion channels that open and close dependent upon the membrane potential [Hodgkin and Huxley \(1952\)](#). This model was relatively accurate, but computationally complex. Simplified neuron models have since attempted to extract the essential components of neuronal dynamics. These have included the Izhikevich model [Izhikevich \(2003\)](#) and the FitzHugh-Nagumo model [FitzHugh \(1961\)](#) which reduced the modelling to just two differential equations. The leaky integrate and fire model is a popular simplified model that places emphasis on the leakage of charge and decrease in membrane potential in the absence of input [Gerstner and Kistler \(2002\)](#), as well as the thresholded generation of action potentials.

2.1.1 Time Encoding of Information using Spikes

Individual neurons are able to encode information in the form of *spike* trains over time. Spikes, also known as action potentials, are the short-lived electrical events involving impulses that are emitted by neurons down their axons, in response to their membrane potential increasing to a level that crosses a certain threshold. Spikes form an essential part of the ability of systems of neurons to process the real-world spatio-temporal data in real time, and in an energy efficient way. There are two overarching ways in which information can be encoded using trains of spikes: these respective schemes are referred to as temporal coding and rate coding [Dayan and Abbott \(2001\)](#). In this thesis, we will make use of the terms events, spikes, and action-potentials interchangeably.

Frequency coding involves the modulation of the frequency of events in an event-based system. It is robust to errors in the precise timing of events, so long as the mean averaged rate remains the same [Izhikevich \(2004\)](#). Temporal coding, in contrast, uses information about the inter-arrival times of events to encode information. Temporal coding can be a much more information-dense encoding method, but may be more prone to interference as a result of delays in events [Butts et al. \(2007\)](#), but may be necessary in scenarios where the conveyance of information over short periods of time is essential, such as sound localisation in the brain, which occurs on the order of milliseconds [Theunissen and Miller \(1995\)](#).

From an information-theoretic perspective, it has been shown that the information theoretic information content of a stochastic point process can be decomposed into two terms: the total number of events, and the times of the events [McFadden \(1965\)](#). Respectively, these correspond to frequency and temporal coding.

Other forms of neural encoding of information may also be used to model the encoding of information in the brain. Bursting [Chacron et al. \(2004\)](#) involves short-lived periods of intense spiking activity; population coding [Georgopoulos et al. \(1986\)](#) involves collaboration between multiple neurons to collectively encode information in time; and phase coding uses the relative timing of spikes to a global oscillatory signal to encode information [Montemurro et al. \(2008\)](#).

2.1.2 Spiking Nonlinearity

The use of nonlinearities in artificial neural networks has been shown to be crucial to allow them to perform universal function approximation. Early on, the Kolmogorov-Arnold representation theorem demonstrated that continuous functions of multiple variables could be approximated using only summation and nonlinear, continuously increasing functions of single variables [Mandic and Chambers \(2001\)](#). Cybenko showed that a single-layer feedforward network with sigmoidal nonlinearities could approximate any continuous function, on a bounded input domain, to an arbitrary degree of accuracy [Cybenko \(1989\)](#). The universal function approximation property was also shown to be valid more generally for multilayer feedforward networks using a broader class of nonlinearities by Hornik et al. [Hornik et al. \(1989\)](#).

The nonlinearity for neurons in the brain is thresholded spiking: a neuron will spike only if its neuron membrane potential rises to a high enough level [Dayan and Abbott \(2001\)](#). Below

this level, no spikes are generated. The repeated stimulation of the post-synaptic neuron by pre-synaptic neurons can produce a number of spikes that diminishes over time, introducing further nonlinearity to the response. This fundamental idea - that nonlinearity is not only present, but essential, for information processing - motivates the use of other types of nonlinear behaviour for processing.

2.1.3 *Synaptic Weight Strengthening and Decay*

In order for neurons to communicate information amongst one another, connections need to exist between them. Such connections in the brain take the form of synapses, which are bridges between the axons (transmitters) of pre-synaptic neurons and the dendrites (receivers) of post-synaptic neurons. One essential aspect of information processing in systems of neurons is the principle of the strengthening and weakening of these connections. There are two main types of potentiation: short term, and long term. Short Term Potentiation (STP) is the short-lived strengthening of synaptic connections. Long Term Potentiation (LTP) is the phenomenon of longer term modification to the strength of synaptic connectivity between neurons Purves (2018).

One mechanism for STP is the decay of weights over time if they are not actively reinforced and strengthened Purves (2018). The weakening of weights represents the forgetting of short-lived connections. The importance of the decay of the weights over time can be understood from a nonlinear system perspective, where the introduction of autonomous state evolution breaks an equivalence to Wiener models and results in an increased nonlinear processing capacity of networks of such components Carbajal et al. (2015). This is yet another case of nonlinear behaviour expanding a system's capabilities for computation.

2.1.4 *Learning Rules: Spike Timing Dependent Plasticity and Frequency Dependent Potentiation*

Spike Timing Dependent Plasticity (STDP) is a proposed local learning rule for biological neurons. STDP - also known as Hebb's rule Hebb (1949) - can be intuitively understood as the notion that "neurons that fire together, wire together". That is, correlation between the firing of a pre-synaptic neuron and the subsequent firing of a post-synaptic neuron induce the strengthening of the synaptic connection. The corresponding anti-STDP states that if there is a lack of correlation, then the connections are actively weakened Purves (2018). It has also been shown that the frequency of the input signal applied to a neuron may influence the degree of strengthening of the synaptic weights - frequency-dependent LTP Purves (2018). We will show how frequency-dependent LTP can be brought about in hardware in Chapter 3.

2.1.5 *Hebbian Competition and Lateral Inhibition*

Hebbian competition refers to the strengthening and weakening of connections relative to the responses of neighbouring neurons. STDP could serve as a passive mechanism to enable this, through competition between synaptic inputs, with stronger signals from presynaptic neurons

inducing spikes in a postsynaptic neuron, resulting in the strengthening of their connection to the postsynaptic neuron, while simultaneously causing other presynaptic spikes to become uncorrelated with postsynaptic spiking, resulting in the corresponding weakening of their connections [Purves \(2018\)](#). It has been shown that competitive Hebbian learning can induce the specialisation of neuronal responses to inputs, and has benefits for information compression [White \(1992\)](#). Alternatively, neurons may actively compete with one another. Lateral inhibition is one form of competition involving a neuron excited by a stimulus actively inhibiting the excitation of its neighbouring neurons. Lateral inhibition occurs in various neurons throughout the brain and can aid in the enhancement of information contrast in local areas [Shamma \(1985\)](#). Competition between neurons can have a wide variety of other benefits for computation, such as playing a role in the formation of memories [Han et al. \(2007\)](#); [Stern \(2007\)](#). Competition is a broader concept that has been used to optimise and fine-tune the performance of physical systems through evolution. Artificial neural networks making use of the notion of competition in a game theoretic sense have shown the ability to model arbitrary data distributions in Generative Adversarial Networks (GANs) [Goodfellow et al. \(2014\)](#). In Chapter 6, we will look at how competition can be used to induce distribution matching in autoencoders, through competition in a minimax communication game.

2.2 MEMRISTORS

Memristors are passive electronic devices that have been proposed as neuromorphic computing components, due to their ability to co-locate the storage and processing of information, circumventing the so-called von Neumann bottleneck. Memristive devices also exhibit a wide range of other nonlinear dynamics that authors have proposed to use and exploit for neuromorphic computing. Chua & Kang originally proposed the use of memristive systems to model neuron dynamics, suggesting that neurons themselves were memristive systems.

In this section, we begin by defining the memristor and detailing the history of the term “memristor” to refer to resistive switching devices. We then give an overview of the memristive device types that have been proposed. We look at the types of noise and nonlinearity that exist in memristive devices, and finally introduce attempts at performing neuromorphic computing using memristors.

2.2.1 Overview and History

Memristors were formulated by Chua in 1971 as the fourth fundamental passive circuit element, linking the elementary electronic variables of the charge and the flux [Chua \(1971\)](#). They completed a theoretical “periodic table” of the passive electronic devices, each defined as connecting two of the four electronic variables, with the other components being capacitors, inductors, and resistors. The original formulation proposed an equation linking the charge to the flux through a (polynomial) relationship, which resulted in an instantaneous resistance determined by the ratio of the voltage to the current, and dependent (in some way) on the instantaneous charge present in the device. The rate of change of the device’s resistive state

was purely a function of the state itself and the magnitude of the voltage applied across the device (or the current flowing into it).

To expand the scope of memristors, Chua & Kang published a paper introducing the notion of memristive systems [Chua and Sung Mo Kang \(1976\)](#). Memristive systems were defined purely mathematically, with no restrictions on the nature of the physical quantities, but only the relationship between them, also allowing for one or more state variables to exist in the device. Despite its generality, this model was unable to capture autonomous state evolution dynamics (later addressed by other models [Carbajal et al. \(2015\)](#)), and was far removed from the physical world, by design, to allow it to be as general as possible, limiting its use for modelling real devices. The term memristor has since come to encompass a great many types of devices.

Years after the theoretical formulation, reasonably reliable resistive switching behaviour was identified and characterised in a titanium dioxide thin film device [Strukov et al. \(2008\)](#), with a claim that the device was in fact a memristor. There is some controversy over whether the device was indeed a true memristor according to the original definition, with claims that the switching did not adhere to the original definition [Vongehr and Meng \(2015\)](#).

Since then, a wide variety of devices and materials have been proposed to effect resistive switching according to different switching mechanisms. prompted a de-facto change in the definition of the term “memristor”, which was fuelled by the broadening of the definition by Chua in 2011 [Chua \(2011\)](#). The types of devices that have been described as memristors, commonly grouped according to the switching mechanism, include Phase Change Memory ([PCM](#)), Electrochemical Metallization ([ECM](#)), Valence Change Mechanism ([VCM](#)), Thermochemical Switching Mechanism ([TCM](#)), electronic device types (including Mott insulator and electron trapping devices), magneto-resistive memories (i.e. Spin Transfer Torque ([STT](#)) or spintronic memristors), and biological memristors, among others. The switching mechanism is often correlated with the material type. For example, [VCM](#) and [TCM](#) switching devices are based on the movement of oxygen vacancies through the active layer of the material, while PCM devices are often constructed from a chalcogenide glass such as Germanium Antimony Tellurium ([GST](#)).

The typical structure of a memristive device is an active layer where resistive switching operations take place, sandwiched between two metal electrodes. Such devices are usually referred to as *thin film*. An illustration of the structure of a thin-film memristive device is shown in Figure 2.2. The device is modelled to have at least two states: the high resistance state, and the low resistance state, but may also have many intermediate states, The resistive states may also be metastable, resulting in decay under certain bias or temperature conditions.

The reason that memristors are of interest from a neuromorphic computing perspective is that the idea of resistive switching modulated by applied bias allows for the notion of storage and computation to be co-located. This overcomes a bottleneck associated with the dual storage-processor model for computers, often called the von Neumann architecture [von Neumann \(1993\)](#), where a memory bank is separated from the processor that computes operations on the information stored in the memory bank. The physical separation of these two computing components leads to the so-called von Neumann bottleneck, where inefficiency

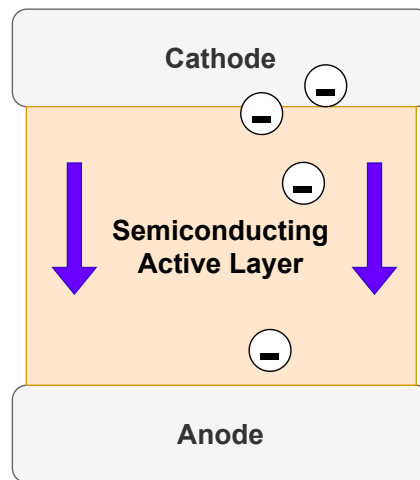


Figure 2.2: A simplified diagram showing the typical structure of a memristive device: a thin layer of semi-conducting or partially insulating material sandwiched between two electrodes. The conductivity of the active layer can be modulated through the use of an applied bias or current through a variety of physical, chemical, or electrical changes.

associated with the transfer of information along relatively long copper wires between the “store” and the “mill” lead to higher energy and time costs for processing.

2.2.2 Types of Memristor

Due to the relatively loose use of the term memristor in the current body of research to refer to such a wide variety of resistive switching devices, it is difficult to provide a definitive hierarchical structure for memristor classification along multiple dimensions. Very often, two devices may bear similarity to each other in one aspect (e.g. their switching mechanism), while being different in another aspect (the materials they are composed of, for example).

Despite this, memristive devices can be separated into a few distinct categories, mostly based on their material composition and switching mechanism, which are often closely connected. These are as follows:

1. Devices based on redox reactions (commonly with a Metal-Insulator-Metal ([MIM](#)) structure)
 - a) Metal-ion based devices ([ECM](#))
 - b) Oxygen-ion based devices ([VCM](#) and [TCM](#))
2. Phase Change Memory
3. Electronic Memristors
 - a) Electron trapping
 - b) Mott insulators
 - c) Ferroelectric memristors
4. Other kinds of memristor

- a) Magnetic (also known as Spintronic or [STT](#)) memristors
- b) Biomemristors

Out of the above types of memristor, by far the most popular are redox memristors, perhaps due to the first device labelled as a memristor being a titanium dioxide thin film device [Strukov et al. \(2008\)](#), as well as PCM. PCM is already a mature technology used for optical storage applications due to its ability to change its optical, as well as resistive, properties during a change of phase. Here, we provide an brief introduction to these two device types.

2.2.2.1 Redox Memristors

Redox memristors are based on a [MIM](#) structure, with the central insulating layer often being referred to as the active layer, since it is the region that modulates the resistance of the device through chemical or structural changes.

For redox memristors, the switching mechanism falls broadly into several categories: interface-type switching, bulk switching, and filamentary switching [Waser and Aono \(2007\)](#). The conduction in such devices may be accelerated by the phenomena of Joule heating, or of nonlinear ionic drift [Strukov and Williams \(2009\)](#).

In Chapter 3, we develop a general model of memristors, focusing on the fit of the model to a resistive drift dataset obtained from titanium dioxide memristors - [VCM](#) devices - whose switching is based on the modulation of the concentration of oxygen vacancies in the active layer of the device. The device is thought to operate via a bipolar, electrically-driven, filamentary switching mechanism [Kwon et al. \(2010\)](#).

SWITCHING MECHANISM [VCM](#) memristors may be classified by their switching: whether they are unipolar or bipolar devices - this refers to whether an applied input signal's polarity influences the direction of switching; as well as whether the mechanism for switching is interfacial, bulk, or filament-based.

Thermally-driven [TCM](#) devices are often unipolar in their switching characteristics, with the magnitude and shape of the applied pulse modulating the switching direction and extent. Other [VCM](#) devices whose switching is electrically driven may be bipolar, with the voltage polarity primarily dictating the direction of switching. Both thermally and electrically driven behaviours can simultaneously exist in the same device, with the dominant behaviour being used to ascribe a category to the device [Jeong et al. \(2007\)](#).

NONLINEAR CONDUCTION AND SWITCHING Joule heating and nonlinear ionic drift may both have an impact on the switching and conduction of redox memristors. Joule heating is the increase in the internal temperature of a resistive device due to the dissipation of power within it. The increased temperature may affect the conduction characteristics of the device may impact switching dynamics and result in accelerated switching [Alexandrov et al. \(2011\)](#). Nonlinear switching behaviour can also be the result of a high field in the short active region of [MIM](#) devices [Strukov and Williams \(2009\)](#).

Although the original formulation of the memristor had an instantaneously linear current-voltage characteristic, for practical devices this may not be the case. Exponential current-voltage

characteristics can arise as a result of diode-like behaviour at the metal-insulator junction [Yang et al. \(2008\)](#), or of conduction through the mechanism of electron tunnelling [Pickett et al. \(2009\)](#).

2.2.2.2 *Phase Change Memory*

PCM has found widespread use in optical storage applications, due to the varying optical properties of the material as the phase of the material changes from an amorphous to a crystalline structure [Gallo and Sebastian \(2020\)](#). The resistance of the device also changes during this phase change and PCM can thus be exploited as a memristive switching medium. The canonical PCM material is GST.

SWITCHING MECHANISM PCM operates by a unipolar switching mechanism. When a current flows through the active layer, Joule heating results in the melting of the layer. Depending on the characteristic of the applied pulse, the device will rapidly cool to an amorphous phase as a result of a short current pulse, or may instead cool more slowly through annealing to a crystalline state for a longer current pulse. Thus, the length and shape of the pulse can be used to modulate the final resistive state of the device [Engel et al. \(2017\)](#).

STATE RELAXATION IN PCM State relaxation is a phenomenon which occurs immediately after programming for PCM devices, involving the dissipation of point defect sites, which serve as electronic “pit-stops” for charge carriers as they conduct charge through the active layer of the device, leading to an increase in the device resistance, due to the decreasing density of local energy wells that enable conduction. State relaxation does not involve a change in the size of the conductive filament or the size of the amorphous state region of PCM devices, in contrast to the mechanism by which resistive drift in redox devices tends to occur, which is associated with the shrinkage of the conductive region or filament. State relaxation can be seen as analogous to the resistive drift phenomenon in redox memristors. This connection will be expanded upon in Chapter 3, where we connect resistive drift in redox devices and state relaxation in PCM, through the use of a volatility state variable [Li et al. \(2012\)](#).

2.2.3 *Noise and Nonlinearity in Memristive Devices*

A range of types of noise, nonlinearity, and nonideality are present in memristive devices. In some cases, the nonidealities of memristors affect their utility in certain contexts and have proved to be challenges if the devices are to be used in a reliable way for storage and neuromorphic applications. In the context of neuromorphic computing, the symmetry of devices could prove important in implementing learning algorithms require potentiation and depression. Additionally, the linearity of devices is important for the emulation of synaptic weights for deep neural network acceleration [Nigun et al. \(2019\)](#). on the other hand, in other cases, memristive nonidealities can in fact be used as computational primitives for neuromorphic computation.

The non-ideal characteristics of memristive devices generally fall into one of several categories. These include:

1. **Programming noise:** Programming noise can arise from the iterative programming procedure needed to change the state of memristive device types such as PCM [Engel et al. \(2017\)](#). It is the inherent uncertainty in the final programmed device state following the application of a particular programming signal [Nandakumar et al. \(2019\)](#).
2. **Read noise:** Read noise is noise that arises during the process of reading the device's state. It can be a result of the read method itself. One important example of read noise is sneak-path current noise. This type of noise comes about as a result of parasitic current paths from the word line to the bit line in a memristive crossbar array [Zidan et al. \(2013\)](#).
3. **Device heterogeneity:** Device heterogeneity is the inherent stochasticity in device characteristics, resulting in inconsistent behaviour among devices of the same type, originating during the manufacture of the devices or developing over time.
4. **Other kinds of material-dependent noise including random telegraph (burst) noise and $1/f$ (flicker) noise:** Low-frequency read noise and random telegraph noise are both types of read noise that have been observed in PCM devices [Nandakumar et al. \(2019\)](#). Random telegraph noise is a type of noise that is present generally in semiconductor devices, consisting of abrupt and random fluctuations in the electrical device characteristics over time.
5. **Resistive drift:** This refers to the tendency of the device state to change autonomously (in the absence of a stimulating input). This phenomenon has been observed in both redox and PCM devices. The cause of this phenomenon may be distinct in different materials. For metal oxide memristors, the diffusion of oxygen vacancies, or the decay of a partially or fully-formed conductive filament, can cause a change in the device's resistance. In general, resistive drift occurs due to phenomena that initiate some form of switching of the device under zero or small-magnitude bias conditions. In metal oxide devices, this can be caused by the slow switching of the device, even under zero-bias conditions. For PCM, structural relaxation can cause the programmed resistance to increase over time, as mentioned in Section 2.2.2.2 above. In addition to structural relaxation, PCM exhibits metastable behaviour following partial programming. It is possible that certain intermediate states may be less stable than the typical binary SET and RESET states. For PCM devices, it has been demonstrated that states will tend to decay to the crystalline state if partial SET or RESET are performed, either due to the growth of a partially formed conductive filament after a partial SET operation, or due to the nucleation of the amorphous capping region formed during a partial RESET operation. The restoration of the crystalline state will be much faster in the case of the partial SET operation, due to the pre-existence of a partially formed conductive filament in the amorphous bulk [Mantegazza et al. \(2010\)](#). In Chapter 5, we take a particular focus on the modelling of resistive drift, while in Chapter 5, we look at the problem of information storage in the presence of this phenomenon.

2.2.4 Neuromorphic Computing with Memristors

The ability of memristors to co-locate the storage and processing of information is the key motivating factor for their use in neuromorphic computing and underpins their ability to emulate the information storage and processing of biological neurons, enabling large scale parallel processing and circumvention of the so-called von Neumann bottleneck¹ von Neumann (1993). Outside of this principle, memristors also exhibit, or can be engineered to exhibit, a range of phenomena which are similar to those present in biological neurons.

Memristors can be used in multiple contexts for neuromorphic computing. One popular proposed use case is their use as deep neural network accelerators, due to their lower power operation Mehonic et al. (2020). In this scenario, memristors are used as low power accelerators for trained deep neural networks, storing the trained weights and executing the addition and multiplication operations necessary for matrix operations through the use of Kirchoff's current law and Ohm's law, respectively, in a memristive crossbar array. The problem of noisy inference due to the nonidealities and noise associated with transferring floating point weights into arrays of memristive devices has been approached from multiple perspectives, including noisy training to improve robustness Oh et al. (2019); Zhou et al. (2020). Some have also noted that the noise introduced to neural networks for noisy training approaches could be of benefit their robustness to adversarial attacks Qin and Vucinic (2018). The benefits of noise for robustness and generalisation will be introduced in Section 2.6.

Memristors, and their nonlinear and noisy dynamics, have also been used for the implementation of a range of the neuronal computational primitives described in Section 2.1, for use in artificial spiking neural networks Kim et al. (2021); Mehonic et al. (2020). The benefits of noise and nonlinearity in memristors were also studied by Carbajal et al. Carbajal et al. (2015), who concluded that heterogeneity in device characteristics could prove beneficial for reservoir computing. The phenomenon of resistive drift, a form of autonomous state evolution, can be considered analogous to the decay of synaptic weights in neurons, as we will discuss in Chapter 3.

The ability of certain kinds of memristor to emulate action potentials has been explored in some works, optionally making use of external circuitry to enable the behaviour Mehonic and Kenyon (2016). Though the emulation of action potentials has been shown as possible in principle empirically, this does not necessarily provide principled methods for the replication of such behaviour in arbitrary devices. In Chapter 3, we build an event-based modelling strategy that can be used to better understand the self-limiting dynamics that can give rise to memristive spiking, as well as the device parameters necessary to enable this, and other neuromorphic behaviours.

¹ The von Neumann bottleneck refers to the inefficiencies caused by the need to transfer information to and from the storage and the processor of a computer, in order to perform computation. The transfer of information can often be the most time consuming and power hungry operation, thus the elimination of this bottleneck offers the potential for huge savings in terms of energy and time.

2.3 MEMRISTOR MODELLING

A key part of better understanding the capabilities and limitations of memristors for the purposes of information storage and processing is the ability to build accurate models of the devices. In this thesis, we introduce two new approaches to memristor modelling. The first is motivated by the need for a more principled quantification of the conditions that lead to the ability of a memristive device to implement the neuronal computational primitives presented in Section 2.1. This approach is presented in Chapter 3. The second approach to modelling stems from the need for a computationally efficient, general, differentiable model for the noisy and nonlinear phenomena present in memristors, in particular the change of information over time in the devices (the resistive drift). We introduce the second modelling approach in Chapter 4.

There already exist a wide variety of models for memristors, however, many of these models take a focus on data storage applications, rather than the neuromorphic setting. A memristor model attempts to characterise one or more of the three characteristics of memristive devices:

1. Switching
2. Conduction
3. Nonidealities

It is essential to be able to produce accurate and efficient models of the physical devices for several reasons:

1. **Experimentation:** The creation of memristor prototype devices can be expensive and time consuming, and thus repeated experiments can be costly to run.
2. **Understanding Device Operation:** Microscopic phenomena which are difficult to observe in-situ during the resistive switching operation, can be better explored and examined through the use of modelling and comparison to existing experimental results, testing hypotheses of the internal workings of a variety of types of memristor [Gao et al. \(2021\)](#). Phenomena such as ion migration, in [MIM](#) devices, cannot be observed directly and must instead be modelled.
3. **Testing Impossible Configurations:** It may also be difficult to test memristive configurations that rely on the future maturation of the technology (e.g. improved reliability). Modelling approaches can simulate behaviour under hypothetical conditions to motivate the development of new device technologies.

2.3.1 Modelling Considerations

Different approaches to modelling are taken, depending on the kind of device being modelled and requirements for the model. For models of physical systems, there are three key factors that can be used to measure the usefulness of a model, depending on the use case:

1. **Physical realism** is emphasised in models based upon realistic physical models, such as that of Strukov et al. [Strukov et al. \(2008\)](#) and Pickett et al. [Pickett et al. \(2009\)](#).
2. **Generality** can be achieved through the use of empirical models (fit to the data), or through using models that attempt to capture the key properties of memristive devices and are agnostic of the particular implementation, such as Chua & Kang's memristive system equations [Chua and Sung Mo Kang \(1976\)](#). Empirical models include exponential models formed from directly observing switching characteristics without reference to the underlying switching mechanisms, of which several examples exist [Stathopoulos et al. \(2018\)](#); [Yang et al. \(2008\)](#).
3. **Computational complexity** is a key consideration and is generally inversely correlated to the physical realism, and also related to the generality.

We take a focus on modelling approaches for metal oxide memristors and PCM devices, due to their widespread use as memristors. Some key characteristics of existing models of these devices include whether they account for the existence of multiple states, as well as how they deal with boundary conditions.

2.3.1.1 *State Multiplicity*

The number of device states is a key modelling consideration. Many authors model memristive devices as binary. Jo et al. develop a model with two possible states of high or low resistance [Jo et al. \(2009\)](#). In contrast, the Generalised Metastable Switch Model ([GMSM](#)) [Molter and Nugent \(2016\)](#) incorporates the possibility for multiple states through the mechanism of multiple stable switches, or a continuous approximation allowing for continuous states. Other models have also been proposed that consider the possibility for multiple stable states [Malik et al. \(2022\)](#). Memristive devices with multiple stable states have a range of applications, so it is important to incorporate this when modelling the devices [Wang et al. \(2022\)](#).

2.3.1.2 *Window Functions and Boundary Characteristics*

A wide variety of early models of memristors make use of a so-called “window function” in order to address problems with the behaviour of the memristive state variable as it reaches the upper or lower permissible limits of the device state. Strukov et al., for example, use a windowing function in their model [Strukov et al. \(2008\)](#), which is $w(D - w)/D$, where w is the state variable (width of the oxygen depleted region, measuring the extent of the hypothesised drift of an oxygen vacancy front through the device), and D is the maximum extent of drift of this front (the width of the device). Note that this results in quadratically diminishing drift as we approach $w = 0$ or $w = D$. Other window functions may be used to simulate nonlinear dopant drift. Prodromakis et. al. supply an overview of various windowing functions [Prodromakis et al. \(2011\)](#) in early memristor models and propose potential solutions. Zha et. al. discuss the “boundary locking problem” [Zha et al. \(2016\)](#) that remains in Prodromakis et. al.'s model and propose modification by including the sign of the current in the window function. The need for a window function is related to the timestepped simulation of the switching of

the memristive state variable. In Chapter 3, we introduce an event-based modelling technique that alleviates the possibility for the violation of boundary conditions entirely, by removing the need for the discrete time approximations of non-event-based models.

2.3.2 *Modelling Nonideality and Stochasticity*

Despite the stochasticity inherent in memristive devices and systems involving the devices, many models of memristors are deterministic. However, some authors do incorporate stochasticity as a fundamental modelling consideration. Jo et al. demonstrate that stochasticity is bias and temperature dependent [Jo et al. \(2009\)](#). Malik et al. develop a stochastic model of memristive devices, based on Markov transitions [Malik et al. \(2022\)](#). The [GMSM](#) consists of a number of metastable switches that stochastically transition from the high state to the low state or vice versa, and collectively determine the device state [Molter and Nugent \(2016\)](#); [Nugent and Molter \(2014\)](#), with a Boltzmann-like switching characteristic. Molter & Nugent note that switching is often dictated by a sinh switching characteristic [Molter and Nugent \(2016\)](#). We will provide a simple, first principles approach to deriving sinh switching as we attempt to improve upon the general modelling approach of the [GMSM](#) in Chapter 3.

2.3.2.1 *Autonomous State Evolution*

Resistive drift (the autonomous evolution of the resistive state), is a phenomenon that is important to include in models of memristive devices. In cases where long-term storage is desirable, volatility may be something we want minimise (see Section 2.4.2). The notion of autonomous state evolution as a modelling feature was explored by Carbajal et al [Carbajal et al. \(2015\)](#), through the introduction of an autonomous term to the state update equation, for example by introducing a leak to the model of a hydraulic memristor [Biolek et al. \(2012\)](#).

There have been attempts to model the time and temperature dependence of structural relaxation in PCM devices [Ielmini et al. \(2007\)](#). It was found that resistance evolved according to the power law $R = R_0(\frac{t}{t_0})^\nu$ where R_0 is the initial resistance and t_0 is a normalisation constant. ν is a constant that depends on the temperature and the read current regime and is termed the drift exponent. This constant also appears in models of defect decay, demonstrating the close dependence between the defect density and the resistance [Li et al. \(2012\)](#). Deterministic models of resistive drift in metal oxide devices, such as titanium dioxide, have made use of a (parameterised) stretched exponential function for modelling the deterministic change of the device resistance over time [Abbey et al. \(2022\)](#); [Chang et al. \(2011\)](#).

We will demonstrate in Chapter 3 that for neuromorphic computing applications, the additional dynamics provided by volatility can help us enable the recreation of neural computational primitives using memristors. In Chapter 5, we will investigate a trade-off between energy and the information storage capabilities of memristors affected by resistive drift.

2.3.3 Data Driven Approaches

Although developing physical models rooted in an understanding of real device mechanics and physics is desirable, accurately modelling the dynamics of a highly complex system may be difficult, or alternatively the simulation of the model may be computationally prohibitive. In addition, physical models may be highly specific to particular device materials and architectures.

The benefits of using a data-driven modelling approach stem from its generality and ability to model a wide range of device types and materials. In addition to this, probabilistic model behaviours and uncertainty can be quantified. An approach to the modelling of the stochasticity present in the programming noise of PCM devices using a data-driven Gaussian kernel model was presented by Engel et al. [Engel et al. \(2017\)](#). The benefits of such an approach included the differentiable nature of the model and the relative computational simplicity of the modelling approach. However, the parameterised probabilistic model chosen restricted the range of conditional distributions that could be modelled. Such an approach also requires a relatively dense sampling of the support of the input distribution to produce an accurate conditional model.

In Chapter 3, we introduce a model of memristive devices specifically tailored to event-driven simulations and neuromorphic applications, taking into account the stochasticity present through use of a Boltzmann probability distribution for switching. To the best of our knowledge, this is the first model to attempt to model memristors in a manner suitable for use in event-driven systems. Following this, in Chapter 4, we investigate the modelling of the phenomenon of resistive drift using a data-driven approach through the use of generative models (see Section 2.5), allowing for an improved approach to probabilistic modelling of the resistive drift phenomenon in memristors.

2.4 SOURCE AND CHANNEL CODING

Whenever we consider the problem of computing in the presence of noise, or dealing with distortions and information loss, a useful framework for thinking comes from communication theory. Communication theory, itself based heavily on information theory, gives us limits and bounds on the rates of compression and transmission of information over noisy communication channels (conditional probability distributions).

As such, we can make use of communication theoretic and information theoretic approaches to store and process information in noisy physical systems. Theoretical tools in these fields aid us in better understanding the limits of noisy information storage, communication and processing. In this section, we introduce the fundamental concepts of source and channel coding, as well as the approach of Deep Joint Source-Channel Coding ([Deep JSCC](#)), which uses deep learning to perform the encoding and decoding of information in the presence of noise. We introduce these topics to motivate approaches to the storage of information in the presence of noise in Chapter 5 and the use of a communication theoretic game and the

exploitation the problem of creating adversarial noise as a solution to a generative modelling distribution matching problem in Chapter 6.

The problem of communicating information is inextricably connected to the problem of learning probability distributions and performing inference. Information exists in the real world, and the problem of understanding the world and forming a model of the deterministic parts of it, such that we are only left to determine those parts that are uncertain and unpredictable from the past, is the same as the problem of communication. In communication, we wish to transfer information - through space or time - from one location to another. We wish to do this in the most efficient way possible. The problem was formalised in Claude Shannon's seminal paper [Shannon \(1948\)](#) in the form of information theory.

In this section, we first introduce the fundamental notions of source and channel coding: a framework for understanding the communication of information between a sender and a receiver, according to a distortion measure, which we use in order to better understand how to operate in the presence of noise in passive electronic devices.

In the final part of this section, we examine the notion of adversarial communications and communication games, looking at existing theoretical results for the saddle points of such games, which will provide background for Chapter 6.

2.4.1 Source and Channel Coding

Source and channel coding are functional mappings we apply to information sources (random variables). Source coding is another term for what is often referred to as information compression. Channel coding is the introduction of redundancy, for the purposes of error detection and/or correction when information is distorted (due to transmission over a noisy channel). If both of these mappings (along with the corresponding inverse mappings needed to recover the original information) are performed in one step, this is termed Joint Source-Channel Coding ([JSCC](#)).

The entropy of an information source is the smallest number of bits of *information*² that can be used to perfectly represent the source. It is dependent on the probability distribution of the source. For a random variable representing our source, say X , the entropy can be expressed as $H(X) = \mathbb{E}_{p_X(x)} \left[\log \frac{1}{p_X(x)} \right]$. If we attempt to represent a source using fewer bits than the entropy of the source, then distortions are inevitable. If we have a particular distortion measure, a function $d(x_1, x_2)$, it is possible to express the distortion as a function of the number of bits of information available to us, which we term the rate, r . The minimal rate achievable as a function of an upper bound on the distortion is called the rate-distortion function, and will depend on the distortion measure we choose. The rate-distortion function is characterised as the infimum of the mutual information subject to the constraint that the distortion does not exceed a given value d' , i.e. $\inf_{q_{Y|X}} I(X; Y)$ s.t. $\mathbb{E}_{q_{Y|X}} [d(x, y)] \leq d'$. Any rate greater than or equal to the rate-distortion function, for a particular distortion, is achievable. Any rate below

² Information is a well-defined concept in information theory. In the context of a particular probability distribution, the number of bits of information obtained by finding the value of sample from the distribution, is the number of binary choices that need to be made to convey the identity of a particular point on the support, such that the expected value (average) of the number of choices is minimised over all points on the support of the distribution.

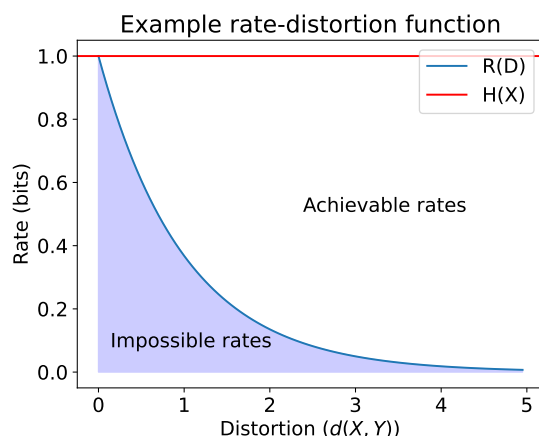


Figure 2.3: An illustrative example of a rate-distortion function for a source with a particular distortion measure, showing the region of achievable rates for given distortions and the entropy of the source as the rate where any given distortion for a source will always be 0. $R(D)$ is the rate-distortion function, yielding the minimum achievable rate R , as a function of an upper bound, D , on the distortion. $H(X)$ is the entropy of the source distribution for the random variable X .

this is impossible to achieve. This is shown in Figure 2.3. Note that once the rate goes above the entropy of the source, a distortion of 0 is always achievable, for *any* given distortion. The distortion-rate function is simply the inverse of the rate-distortion function.

As for the entropy in the context of source coding, the channel capacity is a fundamental limit in the case of channel coding. In the context of transmission of information over a noisy channel, we have a fundamental upper bound on the information we can extract from a noisy random variable about the original, which is given by the mutual information between the original variable and the noisy version. The channel can be equivalently expressed as a conditional probability distribution and an ideal channel code will achieve the upper bound on the rate of transmission (the largest possible mutual information between an input random variable and the random variable at the output of the channel), which we call the capacity of the channel. Achievability of the channel capacity is only guaranteed if we can transmit an infinite number of times before attempting to reconstruct the transmitted symbols [Shannon \(1948\)](#), however capacity-achieving (or close to capacity-achieving) codes do exist for finite blocklengths, for certain channels. The important thing to understand is that the channel forms a bottleneck for information transmission, where no amount of information per transmission greater than the channel capacity can be funnelled through the channel.

Although it was shown that in the infinite blocklength regime, separate source and channel coding of a source are theoretically optimal [Shannon \(1948\)](#), this cannot be guaranteed for finite blocklengths. [JSCC](#) combines the source and channel coding into a single step (and the associated source and channel decoding into another, separate, single step). In this scenario, we optimise the end-to-end distortion, taking into account both the necessary compression and the usage of the channel. A joint source-channel coder must implicitly trade off increased compression of the source bits (and therefore an increase in the lower bound on the achievable distortion) for increased redundancy available for channel coding (and therefore increasing its use of the available channel capacity). In the ideal case, the channel capacity is achieved

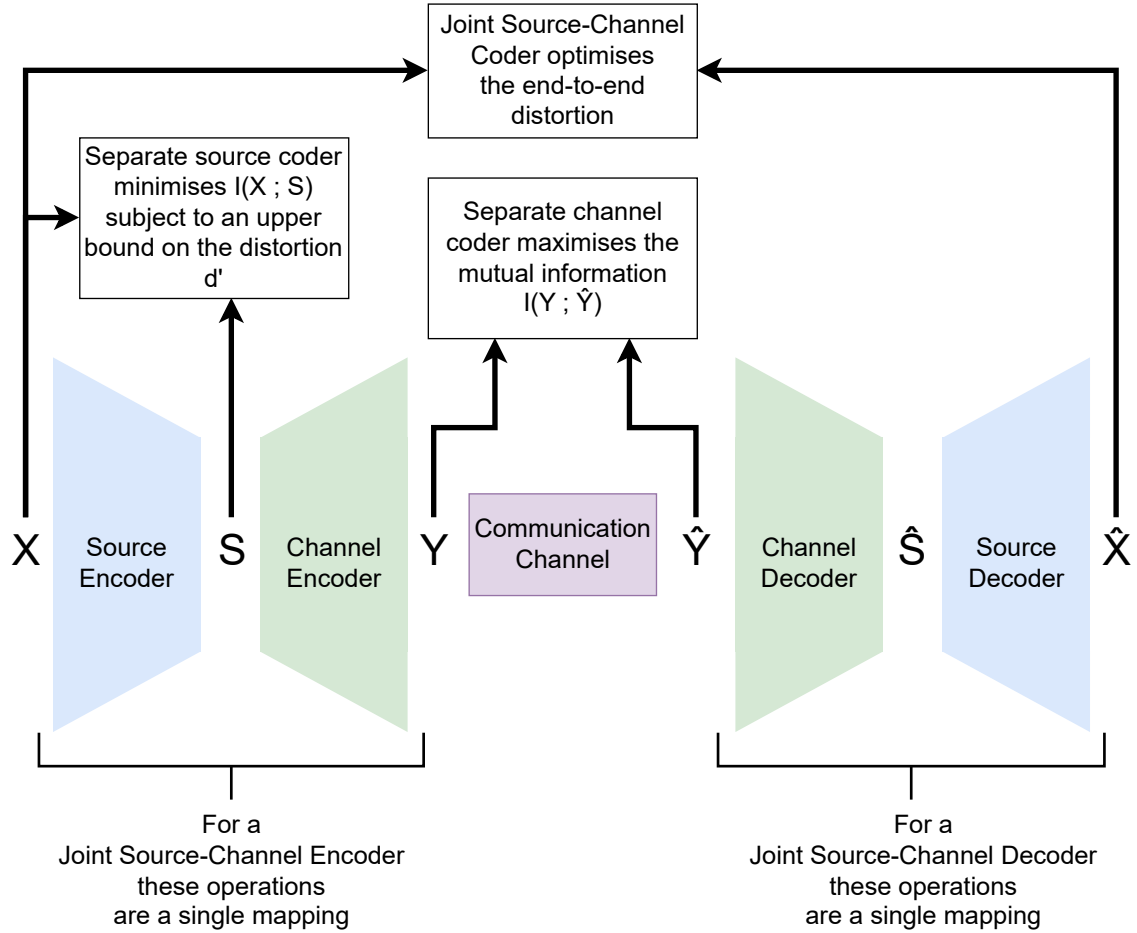


Figure 2.4: A figure demonstrating the roles of source and channel coding. Source coding compresses information, while channel coding introduces redundancy to code against errors. In joint source channel coding, increased compression can be traded off for increased redundancy or vice versa, while maintaining a constant rate of transmission and the source and channel encoders are combined into a single encoder (as are the associated decoders).

by the implicit channel coding step, and - simultaneously - for the rate given by the channel capacity, the minimum distortion (according to the distortion-rate function) is achieved by the implicit source coding. Joint source-channel coding is a superset of separate source and channel coding. A diagram explaining this is shown in Figure 2.4.

2.4.1.1 Deep Joint Source-Channel Coding

Deep JSCC is an approach to JSCC that makes use of deep neural networks for parametrisation of the encoder and decoder networks Boursoulatz et al. (2019). It has been applied to a variety of contexts and modalities, notably the joint compression and transmission of natural images Kurka and Gunduz (2019); Kurka and Gündüz (2020); Tung et al. (2022); Xu et al. (2023). The approach involves the use of an autoencoder, consisting of an encoder and decoder, with a compression rate that is given by the ratio of the input dimension to the dimension of the latent space. Techniques for conditioning the encoder and decoder on the channel characteristics, namely the channel Signal to Noise Ratio (SNR), have included the use of attention modules Xu et al. (2021). In an early formulation of a Deep JSCC autoencoder for

image transmission [Bourtsoulatz et al. \(2019\)](#), the use of a Generalised Divisive Normalisation (GDN) transformation was used to improve the compression of images [Ballé et al. \(2016, 2018\)](#). In Chapter 5, we will make use of and extend this technique for the problem of information storage in the presence of drift on memristive devices.

2.4.2 *Information Theoretic Approaches for Addressing Nonidealities in Memristive Devices*

A communication theoretic approach can aid in the understanding of the noise and nonidealities in memristive devices, as well as the design of encoding and decoding scheme to address them.

The treatment of a storage medium as a communication channel through use of channel coding in the form of error correction codes (and sometimes also incorporation of source coding techniques [Lee \(1999\)](#)) for storing information in traditional communications systems is not a new concept and has been used extensively in traditional storage media. For traditional storage media, there is a trade-off to be made between the longevity/robustness of the stored information and the number of levels used for encoding information. For a storage medium that exhibits degradation over time, such as random access memory, or indeed a memristor, there exists a trade-off between the amount of time we wish our information to be reliably recoverable and the channel capacity of the devices used to store the information, leading to a trade-off between storage density (the number of levels we use to store information) and reliability.

Information theory is essential in the problem of studying the fundamental limits of data storage. As a result, consideration has been made for different kinds of noise in memristive devices and also of the problem of coding against this noise, from an information-theoretic standpoint. This has mostly involved the study of noise that is induced by use of the devices in real circuits, for example sneak path current noise and programming noise. In contrast, in Chapter 5 and Chapter 4, we focus on modelling and addressing drift noise - the noise resulting from autonomous stochastic evolution of the memristive state.

A graph theoretic approach for determining the capacity of binary memristive devices in crossbar arrays in the presence of sneak path currents was conducted by Rumsey [Rumsey \(2019\)](#). The capacity of the arrays for information storage was determined, however practical encoding schemes were difficult to attain due to the need for hand-crafted design. Attempts to solve the problem of decoding information in the presence of sneak path currents have included the design of polar codes and accompanying belief propagation schemes for decoding by using genetic algorithms [Sun, Cai, Song, Quek and Fei \(2021\)](#).

The capacity of PCM memristive devices used in the multi-level storage domain, in the presence of programming noise, has been estimated using discretisation and the use of the Blahut-Arimoto iterative algorithm [Arimoto \(1972\)](#); [Engel et al. \(2015, 2017\)](#). Following this initial analysis, an approach to JSCC for the programming noise distribution, using a Deep JSCC autoencoder, was applied to address the problem of the storage of information [Zarcone et al. \(2018, 2020\)](#); [Zheng et al. \(2018\)](#) using both a statistical model and, subsequently,

in a practical setting using arrays of real devices. These studies also highlighted drift noise as an additional source of imperfection to be addressed.

In the context of information theoretic and communication theoretic approaches to the transmission of information, generative communication channel modelling has been introduced in a number of works, including the use of conditional Generative Adversarial Networks (cGANs) as channel models [Aoudia and Hoydis \(2018\)](#); [O'Shea et al. \(2019\)](#); [Smith and Downey \(2019\)](#); [Ye et al. \(2018, 2020\)](#). In Chapter 4, we extend the use of these techniques to the storage setting, exploring the problem of modelling resistive drift, where the condition (the initial resistance and the delay) are no longer discrete symbols. There, we expand on existing techniques for introducing conditions to GANs [Ding et al. \(2022, 2023\)](#), to address the idiosyncrasies of the problem of resistive drift.

Other approaches to addressing the problems of noise in memristive devices have looked at techniques for enhancing the robustness of neural networks implemented on memristive devices or other noisy analogue hardware. These include noise aware training [Joksas et al. \(2022\)](#); [Qin and Vucinic \(2018\)](#) and model compression and distillation [Isik et al. \(2021\)](#); [Zhou et al. \(2020\)](#).

2.4.3 Communication Games and Worst Case Noise

In this section, in anticipation of our work in Chapter 6, which attempts to make use of adversarial noise to solve a problem in generative modelling, we introduce the idea of an adversarial communication game. This game involves one player - the transmitter - attempting to transmit and recover information over a noisy channel, with channel characteristics that are partially controlled by another player - the adversary. The adversary attempts to minimise the same objective that the transmitter is trying to maximise, hence the problem takes the form of a so-called minimax game.

A very important channel type in the field of communications is the Additive White Gaussian Noise (AWGN). The AWGN channel is a communication channel defined as $c(x) = x + Z$, where $Z \sim \mathcal{N}(0, \sigma_Z^2)$ is a normally distributed random variable with mean 0 and variance σ_Z^2 . The channel belongs to the class of additive noise channels, which are more generally channels of the form $c(x) = x + Z$, with an arbitrary Z , independent of the channel input.

It is well known that the capacity of the AWGN channel is equal to $\frac{1}{2} \log(1 + \text{SNR})$, where SNR is the SNR of the channel³ [Cover and Thomas \(2006\)](#); [Shomorony and Avestimehr \(2013\)](#). If we have a random variable Z such that $\text{Var}[Z] = \sigma_Z^2$, the worst-case (minimum capacity) additive noise channel is the AWGN channel. This stems from the fact that the Gaussian distribution is the distribution that maximises the (differential) entropy for a distribution with a fixed variance.

The principle of adversarial channels, and a minimax game involving a communicator and an adversary who attempts to disrupt communication through the channel, has been previously analysed theoretically in a number of works [Jose and Kulkarni \(2020\)](#). In the infinite

³ The SNR is the ratio of the transmitted signal power to the channel noise power.

delay context⁴, Diggavi & Cover have studied the variance-constrained vector channel setting and obtained game theoretic saddle point results [Diggavi and Cover \(2001\)](#). They addressed the mutual information game between an additive noise channel with a constraint on the covariance matrix and a signalling scheme also satisfying a covariance matrix constraint. The authors demonstrated that the Gaussian transmission and Gaussian noise pair was a saddle point. Another interesting result in this context was shown by Lapidoth, who examined the optimality of Gaussian codebooks and a Gaussian-noise optimised decoder (nearest neighbour decoder) [Lapidoth \(1996\)](#), demonstrating that a Gaussian codebook achieved the capacity of a Gaussian noise channel in an adversarial additive noise setting, regardless of the distribution of the noise, due to the exact cancellation of the positive effect of an increased capacity of a non-Gaussian additive noise channel, and the negative impact of the mismatch between the receiver and the channel characteristics.

In the context of the zero-delay communication regime, the problem of adversarial jamming involves a scenario where a jammer, potentially with access to the source symbols of the transmitter, attempts to disrupt the communication between the transmitter and the receiver. It was shown that in the case of constraints on $E[X^2]$, where X represents the source to be transmitted, and $E[Z^2]$, where Z represents the adversarial noise, that there exist saddle points for the mutual information game that are almost surely unique, and will be Gaussian for a Gaussian source. This problem was studied for scalar sources and later extended to the vector channel case [Akyol and Rose \(2013\)](#); [Akyol et al. \(2013\)](#). Due to the fact that it can be practically achieved, the zero-delay jamming result will be of interest to us in Chapter 6, when we investigate a novel game between a transmitter and a jammer as an auxiliary problem to regularise the latent space of an autoencoder, where we will expand upon the implications of these results.

2.5 GENERATIVE MODELS

We introduced the idea that everything was a model in Chapter 1. The problem of modelling the world around us in order to understand it can be understood from the viewpoint of probability and statistics. We can imagine the environment to be a source of evidence, E , and our model of the environment to be given by a hypothesis, H . Bayesian probability theory tells us that modelling is akin to identifying the underlying distribution of the environment. Bayes' Theorem states that the posterior probability of the hypothesis given the evidence, $P(H|E)$ is the likelihood of the evidence under that hypothesis, $P(E|H)$, multiplied by the prior on the hypothesis, $P(H)$ and divided by the marginal likelihood (the total probability of observing the evidence irrespective of the hypothesis), $P(E)$. $P(H)$ is an agnostic starting point for our belief on the likelihood of the hypothesis in the absence of evidence, that is then scaled by $\frac{P(E|H)}{P(E)}$, the weighted probability of the evidence given the hypothesis. Bayes' Theorem is thus:

⁴ When we refer to infinite and zero-delay, we are speaking of the number of channel transmissions must be made in order to guarantee that the result holds. Of course, the infinite delay is not necessarily practically, but it is theoretically interesting and can yield insights and describe asymptotic performance.

$$P(H|E) = \frac{P(E|H)}{P(E)}P(H) \quad (2.1)$$

We can imagine a hypothesis to be equivalent to a distribution, or random variable, itself, with the support being all possible explanations that may be true for any kind of possible evidence. The evidence is simply some observed data - in the context of machine learning: the dataset. The hypothesis also encodes our set of assumptions on the model generating the data. This can be explicit (e.g. constraining the output of the generative model to a range of values), or implicit: for example, the spatial correlations between pixels induced through use of a convolutional model.

2.5.1 *Generative Models as Parameterised Distributions*

Generative modelling is the complex and often challenging problem of producing previously unseen samples from a known or partially known data distribution. Generative modelling has a wide variety of applications in the fields of natural language, speech and audio, and computer vision. One canonical application of generative modelling is the creation of realistic looking images belonging to a particular dataset.

We can look at the problem of learning a generative model as one of learning the distribution from whence the data came. This can be done by using the data we observe to gradually modify an initial guess about the distribution (our prior) to increase the likelihood of the evidence (the data from the dataset) being observed under the given model: an application of Bayes' Theorem. Initially, we can build our prior by encoding the information we know about the dataset beforehand into the model through the use of inductive biases, that modify the prior distribution on the generative model implicitly. Learning a general model from observing a finite amount of evidence is the logical process of induction - thus, an inductive bias is a bias we incorporate into modelling to guide the process of induction by constraining the space of possible models (or distributions). Said a different way, "training" a generative model involves conditioning our biased model on the evidence in the dataset, biasing it towards being more likely to generate data that exists in the dataset.

We can view a generative model as a parameterised distribution (hypothesis), which is conditioned on the value of its source of randomness. Together, both the combination of the model parameters and architecture, and the source distribution used to create the model, impose a distribution on the output of the model. The source of randomness may be any distribution, but a common choice is the use of a diagonal Gaussian random vector, as in the case of a Variational Autoencoder (VAE), or GAN (see below).

2.5.2 *Types of Generative Model*

Generative modelling techniques in the context of deep learning often fall into one of a number of classes Bond-Taylor et al. (2022). These include GANs Goodfellow et al. (2014),

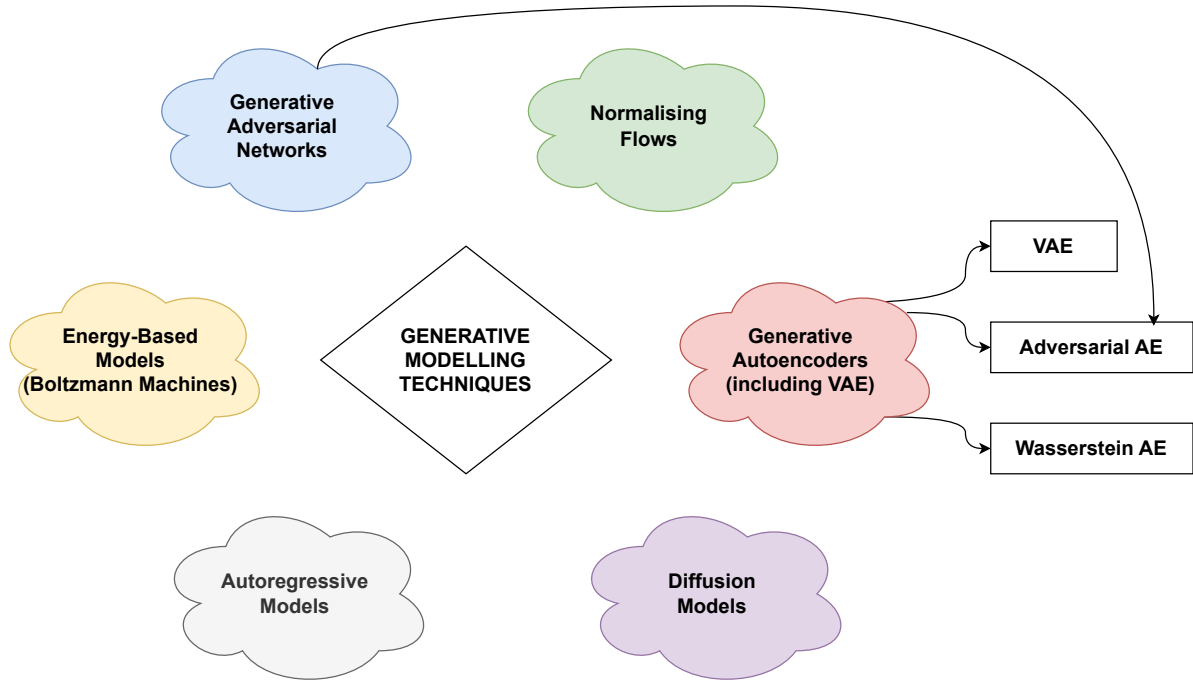


Figure 2.5: Generative modelling techniques fall into a range of categories, seen in this diagram.

VAEs Kingma and Welling (2014), Normalising Flows Dinh et al. (2016), energy-based models such as Boltzmann machines and restricted Boltzmann machines Hinton and Sejnowski (1983), autoregressive models van den Oord et al. (2016), and other hybrid approaches such as Adversarial Autoencoders Makhzani et al. (2016). The broad range of generative modelling approaches is illustrated in Figure 2.5

Here, we look in detail at two types of generative model that are relevant to our research problem: namely, VAEs and GANs.

2.5.2.1 Variational Autoencoders and Generative Autoencoder Models

VAEs use an encoder to condition the prior of the generative model such that it generates an output that is ideally equal to the condition. Conditioning is done by conditioning a proxy distribution - the input of the model - and thus modifying the input distribution.

Among the various techniques for deep generative modelling, VAEs have the significant advantage of not only being able to generate new data samples through sampling from a known prior distribution, but also being able to map samples from the data distribution to a latent space, distributed according to the prior. This mapping can be useful in the context of data compression and of representation learning Bengio et al. (2013). The original VAE formulation relied on maximising the so-called Evidence Lower Bound (ELBO) - a lower bound on the likelihood of generated data coming from the true distribution, used as a proxy for direct maximisation of the likelihood. The formulation can also be viewed as a sum of two terms: a reconstruction loss (dictating sample realism), and an aggregated posterior distribution matching loss term that is the Kullback-Leibler Divergence (KL Divergence) from the diagonal Gaussian prior (dictating the closeness of the aggregated posterior to a specified prior distribution). This second term encouraged the latent distribution to be close to a diagonal

Gaussian distribution, where all elements of the random vector are independent from each other. A number of works have attempted to generalise this formulation by replacing the [KL Divergence](#) term, ($q_Z(z)$), with a different term to encourage the aggregated posterior of the latent representation to match a prior distribution, ($p_Z(z)$). beta-VAE [Fil et al. \(2021\)](#); [Higgins et al. \(2017\)](#) attempted to more explicitly characterise the trade-off between the fidelity of the latent representation to a diagonal Gaussian and the reconstruction quality. Adversarial Autoencoders [Makhzani et al. \(2016\)](#) use a [GAN](#) discriminator that discerns whether the realisations of the latent representation belong to the prior distribution. Wasserstein Autoencoders ([WAEs](#)) proposed using divergences induced by the optimal transport problem, as an alternative to the [KL Divergence](#), for minimising the distance between the aggregated posterior and a diagonal Gaussian distribution [Tolstikhin et al. \(2018\)](#), which can be enforced in a number of ways. These include use of a [GAN](#) to enforce distribution matching in the latent space, a Maximum Mean Discrepancy ([MMD](#)) loss term (with a choice of kernel function e.g. the inverse multi-quadratics ([IMQ](#)) kernel), or the sliced Wasserstein distance [Kolouri et al. \(2018\)](#). Such an approach results in improved sample diversity and quality. Many extensions also exist that aim to improve the generative modelling of [VAEs](#) either through modifications to the architecture or by changing the prior distribution [Sønderby et al. \(2016\)](#); [van den Oord et al. \(2017\)](#).

2.5.2.2 Generative Adversarial Networks

[GANs](#) are a popular type of generative model that consist of two networks that are trained adversarially: a discriminator and a generator. The generator's task is to fool the discriminator into believing that the generator's output stems from the same distribution as a target distribution (dataset). The two networks compete in a minimax game, with the saddle point of the game being such that the Jensen-Shannon divergence between the output distribution induced by the generator, and the ground truth data distribution, is minimised [Goodfellow et al. \(2014\)](#).

Since the conception of the principle behind [GANs](#), there have been numerous developments in addressing the difficulty in training the networks [Arjovsky and Bottou \(2016\)](#). One notable problem that affects the training of [GANs](#) is the problem of mode collapse, which comes about when the generator produces realistic outputs, whose support is only a subset of the true data distribution's. The introduction of Wasserstein [GAN](#), that enforced the minimisation of the Wasserstein distance between the true distribution and that induced by the generator, went some way to addressing the problem of mode collapse [Arjovsky et al. \(2017\)](#). The training of the discriminator using multiple simultaneous samples from the generator has also proved to go some way to wards addressing this problem [Lin et al. \(2020\)](#); [Salimans et al. \(2016\)](#). Many architectural advances have enabled the generation of realistic image data using [GANs](#). Transformer architectures [Vaswani et al. \(2017\)](#) have also enabled better performance in image generation using [GANs](#) [Lee et al. \(2024\)](#).

[GANs](#) are also able to generate information given a condition. This corresponds to the modelling a conditional distribution. The condition could be a categorical label, but it could also be a continuous number. The challenge of conditioning a [GANs](#) on a continuous number has been previously addressed using novel loss functions and label input mechanisms, such as

direct addition of the condition to early latent representations in the network [Ding et al. \(2022, 2023\)](#). In Chapter 5 we will create a [cGAN](#) model for use as a differentiable channel model of the delay and resistance-conditional drift distribution of a memristive device.

2.6 THE USE OF NOISE IN DEEP LEARNING

Noise has been shown to benefit intelligent algorithms in machine learning and deep learning. Both noise and adversarial disruption can be used to improve the performance of deep learning models, from the perspective of robustness and generalisation.

The importance of randomness in deep learning is fundamental. Stochastic Gradient Descent ([SGD](#)) is a cornerstone of optimising neural networks. [SGD](#) was introduced by Robbins & Monro, demonstrating that by incorporating noise in the form of random sampling of training data, [SGD](#) converges more efficiently than deterministic gradient descent approaches. The stochastic nature of [SGD](#) allows for the escape from trapping local minima during the optimisation process, improving learning efficiency [Robbins and Monro \(1951\)](#).

Noisy inputs have been shown to improve the ability of neural networks to correctly classify unseen data. Adversarial robustness is the principle of using adversarial noise to protect against noise injection attacks for the learning of robust representations and to improve model training and has been shown to improve generalisation, partially addressing the vulnerability to adversarial perturbations caused by the linear nature of neural networks [Goodfellow et al. \(2015\)](#).

A number of studies have shown that noise can be used to improve the generalisation of neural networks through robustness. The fundamental technique of dropout for neural networks [Hinton et al. \(2012\)](#); [Srivastava et al. \(2014\)](#), where random elements of the neural network are zeroed out has been shown to improve network generalisation and to reduce the risk of overfitting, improving the robustness of the training procedure by preventing the network from memorising training data samples. This highlights the fact that the principles of information capacity, noise, and generalisation are closely related. A reduced capacity of the network for memorisation results in a reduced likelihood of overfitting.

Researchers have highlighted the potential use of noisy computations during neural network inference [Qin and Vucinic \(2018\)](#). Gal & Ghahramani demonstrated that introducing noise to neural network weights to quantify uncertainty in Bayesian neural networks improved generalisation for a variety of architectures and applications [Gal and Ghahramani \(2016\)](#). In the context of reinforcement learning, the injection of noise into neural network weights can lead to improved strategies for exploration [Haarnoja et al. \(2018\)](#).

Finally, [VAEs](#), introduced in Section 2.5, make use of a reparametrisation trick to enable the differentiable optimisation of sampling from a multivariate Gaussian distribution. This trick can also be seen as the addition of random noise, constraining the channel capacity of the latent space (the information bottleneck [Tishby and Zaslavsky \(2015\)](#)) of the autoencoder, and this view has been used to motivate the development of “variance-constrained autoencoders”, which make use of variance constraints to enforce the Gaussianity of a latent representation through limiting the channel capacity of the latent space and enforcing statistical constraints

Braithwaite et al. (2020). In Chapter 6, we also constrain the variance of the latent space of an autoencoder, but enforce distribution matching in a much more explicit way through the use of an adversarial communication game whose saddle point is a diagonal Gaussian distribution, demonstrating that the generation of adversarial noise can itself be used to regularise the distribution of latent representations.

2.7 SUMMARY

In this chapter, we introduced a range of neuronal computational primitives, and saw the repeated theme of noise and nonlinearity arise in that context. We introduced the notion of memristors as devices used to perform neuromorphic computing and potentially to recreate these primitives artificially. Finally we looked at noise in the context of memristors, and beyond, covering techniques for coding against noise from the perspective of communication and information theory, as well as the potential benefits of noise and adversarial noise for intelligent computing.

Having covered a range of background topics related to the concepts of noise and nonlinearity, and strategies in overcoming them, or exploiting them, in Chapter 3 we take our first steps towards answering the initial research question, by looking at ways in which we can model memristors in a general way for neuromorphic computing applications, and exploring how we can better understand how certain neuronal computational primitives may occur or be brought about in memristors, as a result.

As we saw in the previous chapter, many models exist for memristive devices, with a variety of advantages and disadvantages in terms of their modelling accuracy, generality, and computational complexity. In this chapter, we build a general model of memristors suitable for the simulation of event-based systems, such as hardware spiking neural networks and, more generally, neuromorphic computing systems.

We extend an existing general model of memristors - the Generalised Metastable Switch Model (GMSM) [Molter and Nugent \(2016\)](#) - to an event-driven setting, eliminating errors associated with discrete time approximation, as well as offering potential improvements in terms of computational efficiency for simulation.

To simplify the modelling framework, we introduce the principle of a readout equation, as separate from the evolving memristive state variable, allowing for nonlinear dynamics to be separated from the evolution of the state.

We also introduce a volatility state variable, to allow for the modelling of memory-dependent and dynamic switching behaviour, succinctly capturing and unifying a variety of volatile phenomena present in memristive devices, including state relaxation, structural disruption, Joule heating, and drift acceleration.

Notation We use parameters E_x to denote an energy barrier height in terms of an electron, where $V_x = E_x/q$ is the corresponding equivalent potential difference. We use the notation $\mathcal{X}[i]$ to denote the i 'th element in the ordered set $\mathcal{X}$.

3.1 INTRODUCTION

The modelling of memristive devices is an important component of the development of novel architectures and systems based on these devices, including neuromorphic and event-based computing systems. Models can be classified along several dimensions: their generality, whether they are data-driven or physics based (or both), the inclusion of stochastic or non-ideal behaviour, and suitability for simulation in terms of their computational efficiency (compactness) in the relevant simulation context. The last of these dimensions can be thought of as key - since the goal of a model is to enable the prediction of behaviour so as to allow for the design of strategies to utilise the behaviour of the devices to our advantage. The specific goal of our modelling approach is the development of event-based neuromorphic computing simulations. Accordingly, in this chapter, we aim to extend existing modelling approaches for memristive devices to an event-based setting to allow the models to be used for neuromorphic computing simulations, by introducing general modelling principles, applicable to a broad range of memristive devices.

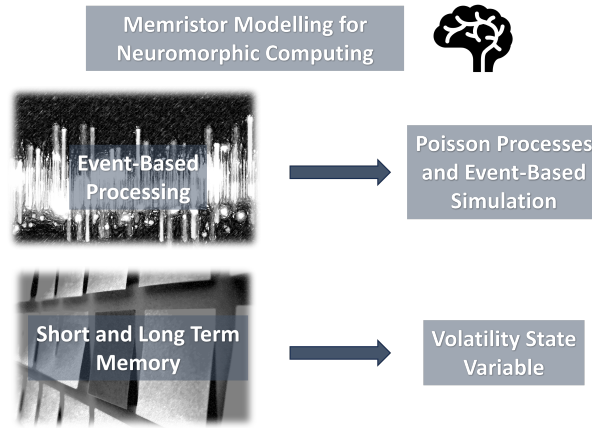


Figure 3.1: A model's use is governed by its predictive ability in a particular setting. We focus on memristor modelling appropriate for use in a neuromorphic computing setting, introducing event-based simulation and the use of a volatility state variable to enable the model to be used to simulate key behaviours necessary for this.

3.1.0.1 Event Based Systems

Event-based systems are systems where changes to the state are driven by the time of events occurring, rather than according to a pre-defined sample time. Such systems can efficiently process information that is sparse in time. Systems that process this kind of information would need to be simulated using finite difference equations, which may be both inefficient due to the need to simulate each and every time-step, and also inaccurate, with inaccuracies that compound over time due to the use of a non-infinitesimal (finite) time-step for simulation.

Our motivation for introducing an event-based simulation framework is to allow for the simulation of memristors in event-based settings, most notably, spiking neural networks implemented on hardware, in the context of the growing research area of neuromorphic computing (computing that aims to use principles from neuroscience to improve the processing of information) Mead (1990). Indeed, the notion of event-based processing is inspired by the function of the human brain. Memristors have been touted as potential neuromorphic computing elements, due to their ability to process information in-place, circumventing the von Neumann bottleneck. This bottleneck refers to the separation of storage and processing and the associated high energy and computation time costs of transferring information from the 'store' (storage) to the 'mill' (processor) and back during computation von Neumann (1993). Circumventing the von Neumann bottleneck offers the potential for massively parallel processing, akin to that possible with biological neurons.

It is clear that in order to implement event-based computation in hardware with memristors, it would be more natural to use models which are suited to event-based simulation settings.

3.1.0.2 Volatility

One key aspect of information processing in the human brain is the split of memory into long-term and short-term, often labelled as Long Term Potentiation (LTP) and Short Term Potentiation (STP). The state variable in the memristor model can capture long-term memory changes, while the possibility for autonomous state evolution captures short-term memory

change to some degree. In order to enrich the model’s ability to describe STP phenomena, we account for the possibility of short term memory outside of the state variable through the inclusion of a volatility state variable. This variable allows for the capture of a variety of physical phenomena in various memristive systems, including Joule heating in thin film metal oxide devices, nonlinear drift in metal oxide devices, and state relaxation in Phase Change Memory (PCM) devices.

Several different phenomena in both metal oxide devices and in PCM rely on the principle of autonomous state evolution. In metal oxide devices, the autonomous degradation of the conductive filaments results in the phenomenon of resistive drift. In PCM, the phenomenon of structural relaxation results in a change of the device’s resistance over time. As a result, we also extend the modelling framework by introducing a volatility state variable, as a unified way of describing the autonomous state evolution behaviour seen in a variety of memristive systems.

3.1.1 *Outline of Chapter*

In Section 3.2, we look at a new simulation paradigm for event-based systems which is natively event-driven, based on the evaluation of time-to-fire for multiple parallel Poisson point processes, while also allowing for continuous-time changes - such as changes in temperature or bias - to be incorporated. We extend an existing generic memristor model, the GMSM Molter and Nugent (2016); Nugent and Molter (2014) to the event-based setting, in order to address problems associated with the time-stepped simulation, where error is inevitable and proportional to the size of the timestep. We modify the rate calculations of the model, to base them on a more physically generic Boltzmann probability density function. We also generalise the model through the separation of the resistance and the state variable via a readout equation.. In addition, we generalise a variety of nonlinear, temporary and autonomous evolution phenomena observed in different types of memristive device - thin film Metal-Insulator-Metal (MIM) devices and PCM - and propose the use of a single variable called the “volatility state variable” to singularly describe all these phenomena.

In Section 3.3, we provide qualitative and quantitative discussions and analysis of the model behaviour. We examine at the characteristics of the model in detail, defining the equilibrium point of the model and exploring the impact of parameters on the simulations.

In Section 3.4, we present a real drift dataset for titanium dioxide memristors. We introduce a linear conductance approximation and physical model for the thresholded switching of the titanium oxide memristor. We then present an approach for fitting the event-based model to the device characteristics, such that the mean drift behaviour matches that of the data and derive the appropriate model parameters for our dataset.

Finally, in Section 3.5, based on our modelling framework, we explore the implementation of several neuronal computational primitives: frequency-dependent potentiation, intensity modulated generation of action potentials, and a discussion of the implementation of nonlinear thresholding achieved through use of the proposed switching characteristics.

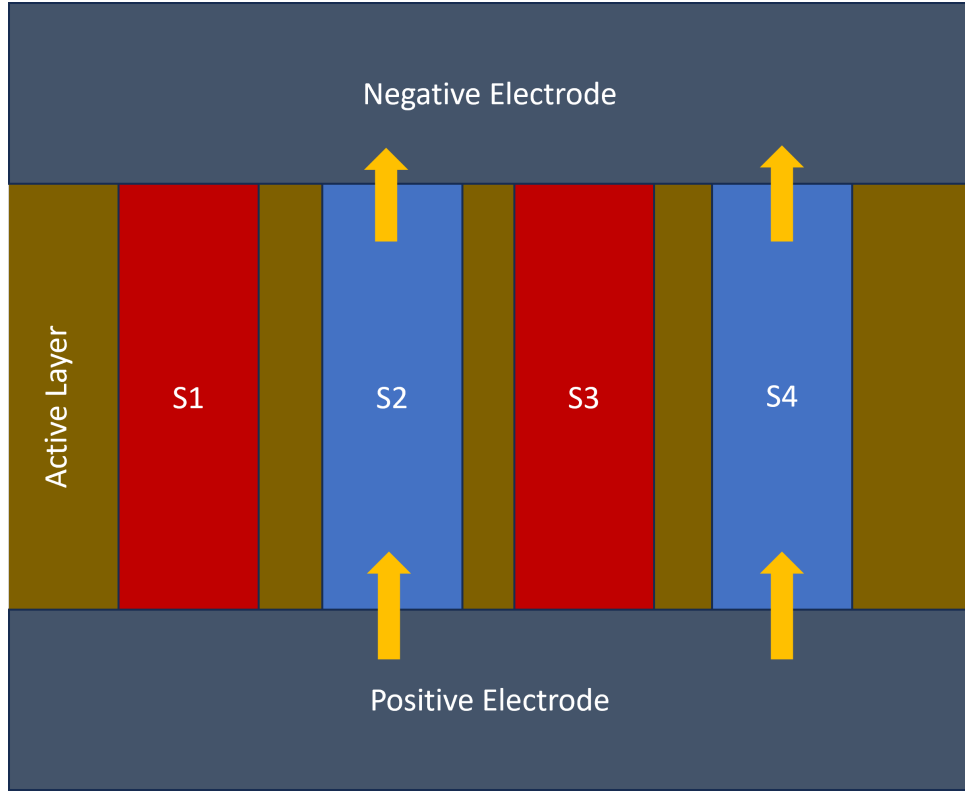


Figure 3.2: The metastable switch model. The conductance is a linear function of the number of switches (parallel conducting channels) that are in the “on” state at any one time.

3.2 EVENT-BASED METASTABLE SWITCHING MODEL

In this section, we introduce the proposed modelling framework. We first propose an extension to the [GMSM Molter and Nugent \(2016\)](#) - a general model for memristive devices - and examine a modified metastable switch model based on continuous time probabilistic point processes, rather than discrete time Bernoulli random variables, in order to make the model event based. We then incorporate the so-called “volatility state variable” to allow for richer modelling of physical phenomena such as autonomous state evolution, Joule heating, and nonlinear drift, which in turn allows for the characterisation of certain short term memory behaviours in memristive devices.

3.2.1 Limitations of the Discrete-Time Stochastic Switching Models

Many models for memristive devices have been proposed; however, only a handful of these take into account the stochasticity inherent in the devices [Malik et al. \(2021\)](#); [Naous et al. \(2016\)](#), with the majority making deterministic assumptions.

One general memristor model that includes stochasticity as a fundamental feature of the model, is Molter & Nugent’s [GMSM Molter and Nugent \(2016\)](#). It models the memristor as a number of parallel metastable switches, each allowed to take one of two different states - the “high resistance” or “low resistance” state (see Figure 3.2), with stochastic transitions occurring every unit time with probabilities that are dependent on the temperature and bias.

The **GMSM** bases its switching mechanism upon a Fermi-Dirac-like activation function (a logistic function), based on the temperature and the bias, which dictates the rate at which switching events occur. Two switching processes are modelled: low resistance to high resistance, and high resistance to low resistance. The probability of a single metastable switch transitioning from the low resistance state to the high resistance state in a given timestep ΔT is denoted by P_A , while the probability of a transition from the high resistance state to a low resistance state in ΔT is denoted by P_B . These two probabilities are given by

$$P_A = \alpha \frac{1}{1 + e^{\beta(V - V_A)}} \quad (3.1)$$

$$P_B = \alpha \frac{1}{1 + e^{\beta(-V + V_B)}} \quad (3.2)$$

where $\beta \triangleq \frac{q}{k_B T}$ is a physical parameter that introduces temperature-dependence and is the inverse of the thermal voltage (k_B is Boltzmann's constant and q is the electronic charge), $\alpha = \frac{\Delta t}{t_c}$ is a ratio of the timestep to the characteristic timescale t_c - a parameter of the device.

The **GMSM** models the device's state evolution over time by considering the number of switches in either the high or the low state and using the respective transition probabilities to define two binomial distributions that can be used to simulate the number of respective transitions of switches from low resistance to high, or vice versa for the second distribution, over a time-step. The normal approximation to a binomial random variable can be used, as N (the number of metastable switches) becomes large, to develop a continuous stochastic model.

We note that the approximation in Equation 3.1 is valid only for small values of Δt . If we let the value of Δt grow large, then errors associated with discretising the timestep begin to emerge. The larger the value of Δt , the more significant a change will have occurred in the parameters used to calculate the transition probabilities, leading to modelling inaccuracies. This is a problem for any discrete-time approximation of a continuous-time problem i.e. any finite difference method for approximating a continuous-time differential equation. Error-free modelling can only be achieved in the limit as the timestep vanishes, i.e., $\Delta t \rightarrow 0$.

In order to address the discrete-time errors associated with the **GMSM**, as well as being motivated by the use of memristors in event-based neuromorphic computing systems, we consider modelling the state transitions not in terms of a transition probability per timestep, but rather through a (Poisson) point process, whose rate parameter dictates the frequency of switching events. This approach switches from a timestep-based evaluation to an event-based evaluation, where we model the changes to the device's state as and when they occur. This removes the discretisation error. It also offers potential benefits from a computational efficiency perspective, since the frequency of model updates are tied to frequency of change of the memristive state.

Next, we specify the model dynamics and modelling principles.

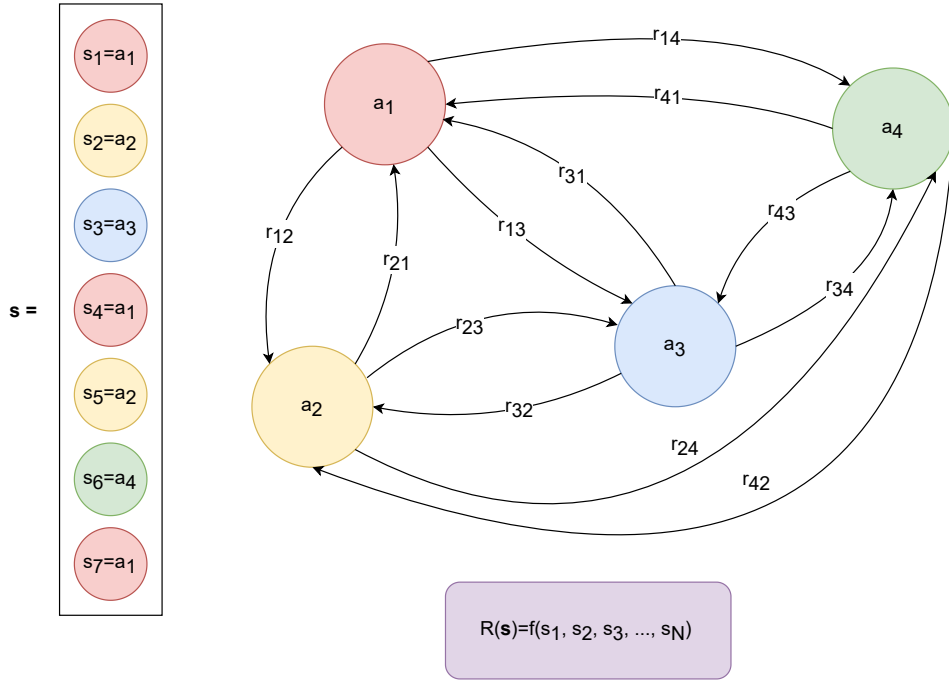


Figure 3.3: A diagram showing an example of the state transitions of a single switch for $k = 4$ states. The readout equation $R(\mathbf{r})$ is a function of the state vector denoting the instantaneous states of all of the metastable switches $s_i, i \in [1 \dots N]$.

3.2.2 State Modelling

We generalise the notion of metastable switches introduced by Molter & Nugent [Molter and Nugent \(2016\)](#) to allow for the possibility of multiple states per switch. We define state variables as variables that influence the relevant readout values, such as the voltage-current characteristic of the device at any point in time, or do not, but impact the rate of switching of variables that do (the case for the volatility state variable, which is described in Section 3.2.5).

Although the [GMSM](#) specifies binary metastable switches, it is also possible to extend the framework to a setting with more than 2 states per variable, multiple state variables¹, or continuous state variables. The [GMSM](#) also requires that the state variable be linearly connected to the conductance, however we generalise this notion in Section 3.2.3.

An example state space is shown in Figure 3.3. The multiplicity of the full state space will be the product of the multiplicities of the independent state spaces of the N state variables, resulting in k^N possible state configurations, where k is the multiplicity of the state space for a single variable.

In keeping with the terminology of the [GMSM](#), we use the terms “switches”, or “metastable switches”, to refer to the state variables of the model (the volatility is separately dealt with). For the purposes of this work, we assume that both the number of state variables (N) and the multiplicity of the individual state spaces are finite. Specifically, we consider the case where $k = 2$ - i.e. binary switching. The extension to the continuous setting is left as future work. We use n to denote the number of switches in the state that is conducive to conduction,

¹ There might be more than one possible choice of state variable, but ideally the variables chosen should present a minimal level of redundancy (high level of independence/disentanglement) and/or be rooted in a physical model.

which we call the “up state”, the “high conductance state”, or the “low resistance state”. Then, $m = N - n$ denotes the number of switches in the state that is not conducive to conduction, which we call the “down state”, the “low conductance state”, or the “high resistance state”. Switching from a lower to a higher resistance state is termed “up” switching, while switching in the opposite direction is termed “down” switching, reflecting the respective increase or decrease in the resistance. Note that due to the deterministic relationship between m and n , there is effectively only one state variable, yet we make use of both for notational convenience.

The physical mechanism motivating the use of N parallel, binary, metastable switches used as a model for memristive devices could be one of the following: the number of oxygen vacancies in the bulk of the material vs. those which are attached to metal ions, where reduction has not occurred; the number of vacancies/ions that are present in the (main) conductive filament, versus those which are not, as per the popular, single dominant conductive filament model; or the number of parallel conducting pathways running through the device, vs. those that are not formed.

We detail the inclusion of an additional state variable that models the volatility in a device, motivated by physical evidence for similar phenomena in multiple resistive switching devices, in Section 3.2.5.1. This additional variable does not affect the readout equation, but may affect the switching rates. The motivation for having a single state variable that impacts the readout equation is that this is in keeping with the spirit of most resistive switching memories, whose state is thought to be closely tied to their instantaneous resistance. In addition, it has been shown that a single state variable is sufficient to explain the behaviour of various memristive devices [Pickett et al. \(2009\)](#); [Strukov et al. \(2008\)](#).

3.2.3 Readout Equation

Note that the voltage-current relationship for a memristive device may not always be instantaneously linear. Indeed, an exponential or sinh relationship is possible depending on the nature of the device being considered [Molter and Nugent \(2016\)](#). Different behaviour can be considered by relating the memristive state variable, n , to a separate quantity such as the parameter for a Shockley diode equation, or the scaling factor for a sinh voltage-current relation.

The simplest approach to modelling a memristor would be to use a state variable that tracked a quantity that is directly linked to the resistance. In the original formulation of the memristor [Chua \(1971\)](#), Chua suggested that the charge (or the flux) be the state variable. The amount of charge or flux that the device possessed could be directly connected to the resistance through transforming the equation linking the charge and flux to an instantaneous relationship between the current and voltage. In the simplest case, the relationship between the charge and flux would be linear and the memristor would simply be a resistor without memory. Polynomial relations that link the charge and flux, on the other hand, would give rise to memristive behaviour (memory of the device resistance) [Chua \(2011\)](#); [Guo et al. \(2015\)](#).

However, real memristive devices do not necessarily adhere to this ideal model. Indeed, the definition of the memristor has evolved far beyond what Chua originally formulated, with

Original Memristor Formulation

Memristive Systems and Beyond

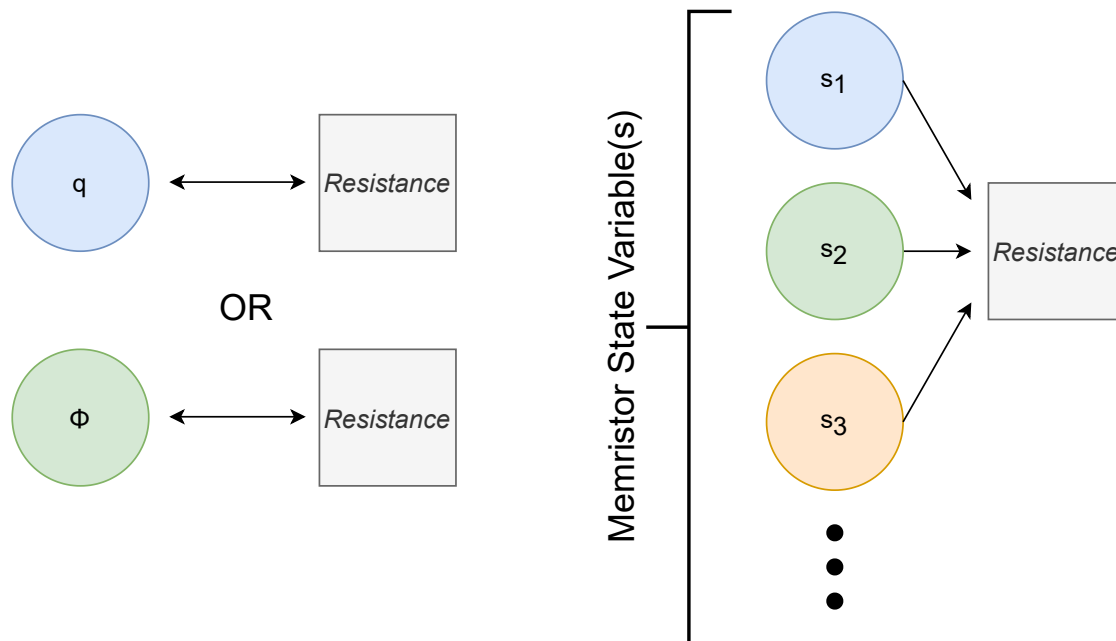


Figure 3.4: According to Chua’s original formulation, there should be a one-to-one relation between either the charge or the flux and the memristor resistance (memristance). For memristive systems, there may be multiple state variables, not necessarily including charge or flux, that determine the memristance, and the state vector and memristance are in a many-to-one relationship (that is, the states cannot necessarily be fully characterised from the memristance alone).

many devices and systems that termed “memristor” in the literature, which would be more correctly described as what are known as “memristive systems” Chua and Sung Mo Kang (1976). This includes a broad class of resistive switching memory devices, including PCM and Resistive Random Access Memory (RRAM)².

Such systems and devices may not be described simply using a single state variable linked to the resistance. Instead, we separate the state variable(s) of the device and view the resistance, or indeed the voltage-current characteristic, as a “readout” (many to one function) of the state variables. Given an appropriate choice of state variables, the history of a device that is relevant to predictions of future readout values can be encapsulated in a fixed number of dimensions. For example, in the case of the original model of the memristor, the history of the device was captured by the charge state variable. In other cases, a higher dimensional representation of the history might be necessary.

Rather than requiring that the resistance be linearly linked to the state variable, we instead use a readout equation to provide flexibility in simulation. The readout equation takes as input the memristive state variable n , and outputs a resistance. This allows for a consistent description of the state, and allows for the separation of measurement errors from the state evolution dynamics.

² Usually this term is used to refer to metal oxide devices, but it technically also encompasses PCM. However, the de facto usage currently means that RRAM is synonymous with thin film metal oxide devices.

In Section 3.4, we see an example of a thresholding, nonlinear readout equation, that links the state variable to the voltage-current characteristics in a nonlinear way.

3.2.4 Switching Mechanism

The **GMSM** models the metastable switching dynamics through a set of N parallel Bernoulli random variables (one for each switch) for each timestep. As noted, our modelling approach differs in that it models the switches as independent Poisson point processes happening in parallel. The N independent Bernoulli random variables are then combined into two binomial random variables - or, in the continuous approximation, a normal distribution - that measure the state change for all switches in a single timestep ΔT .

We instead approach the modelling by replacing the Bernoulli probabilities with rates for the independent Poisson point processes. We use the convention that the subscript *up* refers to parameters describing the up transition (and the same, respectively, for subscript down and down transitions).

The rate parameter for the Poisson process of a single metastable switch is referred to as the base rate and denoted by either λ'_{up} or λ'_{down} . In each case, its value is determined by a Boltzmann activation function (see Section 3.2.4.1 below) that is dependent on the voltage bias and the temperature, as well as static parameters. To obtain the overall rate (the cumulative rate of all up or down switching events), we simply scale the base rate by the number of switches in the origin state - m for λ'_{up} , or n for λ'_{down} , giving us the cumulative up and down switching parameters, λ_{up} and λ_{down} . We use the convention that a positive voltage makes it more probable for the device to switch to a higher resistive state, while a negative voltage makes it more probable for the device to switch to a lower resistive state.

Equation 3.3 below, gives the Poisson rate of a single switch transitioning from the “high” - or “open circuit” - state, m , to the “low” - or “conducting” - state, n : λ'_{down} . Note that in the equations, two physical constants are also used: Boltzmann’s constant, k_B , and the electronic charge, q . As a result, Equation 3.4 gives the rate parameter for the overall up switching process.

$$\lambda'_{up}(t) = \exp\left(-\frac{E_a - kV(t)q/2 - E_{off}/2}{k_B T(t)(1 + \rho(t))}\right) \quad (3.3)$$

$$\lambda_{up}(t) = n(t)\lambda'_{up}(t) \quad (3.4)$$

where the parameters are as follows:

1. $E_a = qV_a$ is the activation energy (the energy barrier height). This increases or decreases both the up and down switching rates.
2. $E_{off} = qV_{off}$ (added, rather than subtracted, for λ'_{down}) is the energy offset that leads to a shift of the equilibrium point from equal numbers of states being in the high or low resistive state. This adjusts the balance of switching rates under zero bias.

3. k is a constant factor used to convert an applied bias to the effective voltage used for determination of the effect voltage and subsequent shift in the barrier height, accounting for material-specific response factors, such as lattice periodicity.
4. $V(t)$ is the voltage across the device at time t , which will be added for λ'_{down} as opposed to being subtracted. This gives an additional energy offset when scaled by q .
5. $T(t)$ is the time-dependent temperature variable in degrees Kelvin. This may be influenced by the history of applied voltages and the device state history, which directly impact the rate of switching, as well as the equilibrium point when E_{off} is not 0.
6. $\rho(t)$ has a similar impact to the temperature and is the volatility state variable, that may also be dependent on the time and voltage/state history.

In this model, application of a voltage bias, $V(t)$, lowers or raises the activation energy barrier in one direction, simultaneously raising or lowering it in the other direction. This is illustrated in Figure 3.5. Note also that, for the purposes of simplicity we assume throughout this chapter, without loss of generality, that $k = 1$.

The volatility state variable is an additional state variable that is introduced to incorporate stateful effects on the evolution of the resistive state, discussed further in Section 3.2.5.

3.2.4.1 Boltzmann Activation Function

To determine the rates of the Poisson processes, we made use of a Boltzmann occupation function, rather than the Fermi-Dirac occupation function (logistic function) used to determine the switching probabilities in the GMSM. We make use of the Boltzmann distribution due to it being a good model of probabilistic behaviour in microscopic thermodynamic systems, modulated by energy barriers.

In the case of a metal oxide memristor, we envision ions hopping over energy barriers, mediated by voltage bias and temperature conditions, resulting in the movement of ions into more or less favourable locations for conduction (modulating the instantaneous resistance). The nature of the ions is not crucial: for example, they could be metal ions (as in Electrochemical Metallization (ECM) devices), or oxygen vacancies (as in Valence Change Mechanism (VCM) devices). There are many possible mechanisms that could drive the movement of ions in an ionic movement memristive device. Filamentary, interfacial, and bulk switching (e.g. Mott insulator-metal transition through ionic doping/trapping) mechanisms have been proposed. Such a physical switching mechanism motivates the use of a Boltzmann modelling approach. The use of sigmoid-like or Boltzmann-like activations is common in models of memristor state evolution, and a sigmoid activation is used in the GMSM simply to bound the probability of state transitions between 0 and 1 Molter and Nugent (2016).

sinh switching behaviour (for an applied voltage) naturally arises out of modelling that uses an energy-barrier assumption with a height and hopping rate that is exponential in the applied voltage. sinh-switching behaviour is something that has been observed in a variety of memristive devices and attributed to phenomena such as exponential ionic drift for high fields, or Joule heating effects Pickett et al. (2009); Strukov and Williams (2009). This can be observed

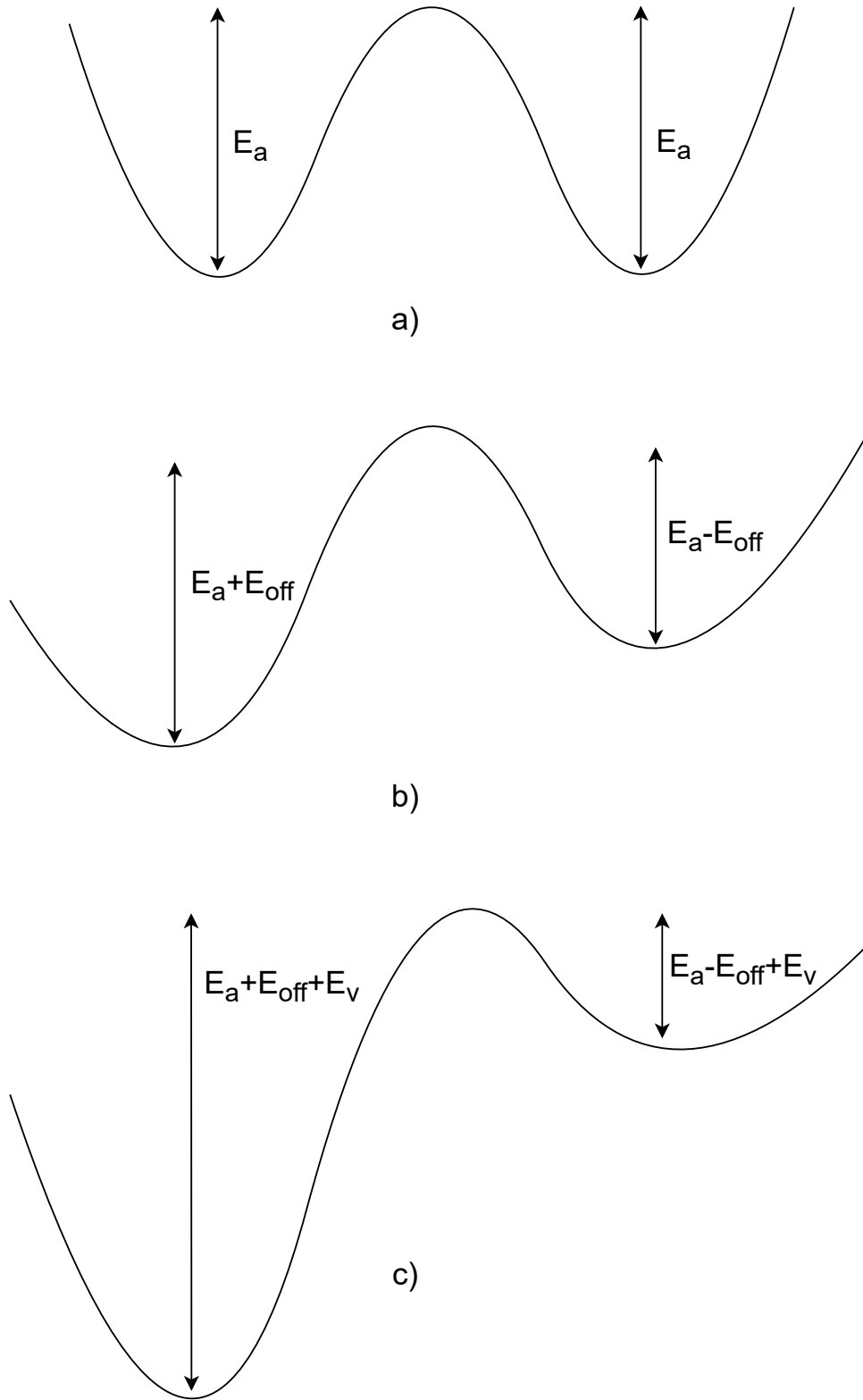


Figure 3.5: Illustration of the effect of the different parameters on the Boltzmann barrier heights in the forward and reverse switching directions. In a) we see the standard barrier, unmodulated by an offset or an applied voltage. b) shows the barrier modified by an energy offset E_{off} . c) shows the energy barrier modulated by the offset in addition to the modulation of the barrier resulting from the application of a voltage bias, E_v . The relative energy barrier height for each local minimum determines the respective Poisson switching rates λ'_{up} and λ'_{down} .

by subtracting the instantaneous rate of upward transitions (transitions from the down state to the up state) from the downward transitions (transitions from the up state to the down state) as follows.

In [Strukov and Williams \(2009\)](#), Strukov & Williams note that for the rigid point ion model, the energy barriers in the forward and reverse directions for the drift of ions in a lattice would be distorted, through the application of an electric field, by $\pm q\varepsilon a/2$, where ε is the field, and a is the periodicity of the lattice. The characteristic length a need not be lattice periodicity, therefore we simply combine the impact of the lattice periodicity, specific to the rigid point ion model, and the electric field into a parameter $V = \varepsilon a$, thus the distortion will be $\pm qV/2$, respectively.

The average energy of any ion will be approximately the thermal voltage $V_T = k_B T$, thus providing a mechanism for thermally accelerated drift, through Joule heating effects.

Based on this, if we assume that the rate of hopping upwards or downwards for each “switch” is given by a rate equation based on the Boltzmann probability density³, then we can say that the rate of transition from the high resistance state to the low resistance state for individual switches is given by:

$$r_{\text{up}} \propto e^{-(E_a - qV/2)/k_B T}$$

while for the transition from the low resistance state to the high resistance state we have:

$$r_{\text{down}} \propto e^{-(E_a + qV/2)/k_B T}$$

In turn, we can see that subtracting the rates gives us the instantaneous, net rate for movement from a high resistance to a low resistance state:

$$r_{\text{net}} \propto e^{-(E_a - qV/2)/k_B T} - e^{-(E_a + qV/2)/k_B T} \quad (3.5)$$

$$\propto e^{-E_a/k_B T} \sinh(-qV/2k_B T) \quad (3.6)$$

This is similar to the equation for high field ionic drift for the rigid point ion model described in [Strukov and Williams \(2009\)](#). However, the net rate is only an instantaneous rate of change. In our model, any event (transition from up to down, or down to up) causes a change in the number of switches in each state, and thus a linear and proportional change to the rates λ_{up} and λ_{down} . As a result, we must re-calculate the rates whenever there is a change in the state.

3.2.5 Volatility State Variable

Memristive systems, such as PCM and metal oxide devices, differ significantly in their structure, material composition, and switching/conduction mechanics, but can nevertheless exhibit similar characteristics. For example, in the case of their switching behaviour under bias, both types of device have associated physical phenomena which cause the device's state

³ The Boltzmann probability density is a common model for the probability density of particles at thermal equilibrium.

to autonomously change under zero bias, following initial programming. This motivates the introduction of a unified way of describing these disparate physical phenomena for the purposes of modelling.

We introduce the notion of “volatility”. The term “volatility” is sometimes used to refer to the phenomenon of resistive drift - the tendency for the resistance of a device to deviate from its initial value over time. **We here generalise the term to include any kind of transient physical phenomenon that can affect the switching or conduction of a memristive device.** Examples of physical phenomena that fall under the umbrella of volatility include:

1. Joule heating resulting in nonlinear switching behaviour and/or accelerated carrier drift under bias in various memristive device types, including metal oxide and PCM devices.
2. Structural relaxation in PCM devices.
3. Structural disruption that causes reconfiguration in diffusive memristive devices.

The presence of volatile mechanics in a device tends to express itself in several ways: memory-dependence in switching and conduction behaviour; self-excitatory or self-inhibitory behaviour during switching and conduction leading to nonlinear switching and/or conduction; and autonomous state evolution as a result of temporary changes within the devices, such as increased temperature or disorder in the material. The unification of the phenomena removes the need to identify the exact cause of memory-dependent switching behaviour in memristive devices, allowing us to focus only on the consequences for the evolution of the state variable. The volatility state variable summarises the device history that is relevant to future switching behaviour into a single state variable.

In the general sense of the word, an increase in volatility is associated with an increase in the state transition probabilities (in the discrete setting). This contrasts with a directed bias, which will result in an increase in one or more switching rates at the expense of decreasing one or more other switching rates.

Intuitively, if we imagine switching events to be associated with a change in the state of the device, an increase (decrease) in volatility means that the number of state change events per unit time increases (decreases). Depending on the degree to which the different switching rates increase, relative to one another, a change in the volatility may also have an impact on the local metastable (equilibrium or capturing) state(s).

Note that our definition of volatility motivates the way in which we model its effect mathematically for a given switching probability model. For example, in later sections we will explore the use of a Boltzmann activation energy model, and the volatility will have a similar effect to the temperature, increasing all switching rates simultaneously.

3.2.5.1 *Phenomena Captured by the Volatility State Variable*

One key observation is that volatility is higher immediately after a programming event in a range of memristor devices, including [ECM](#) and [VCM](#) ionic movement memristors, and in PCM devices. It may also be the case that the degree of volatility is dependent in some way on the nature of the programming (i.e., it is not a memoryless phenomenon given the memristive

state alone), with more extensive programming pulse application or a higher deviation from a previous stable level resulting in a higher degree of volatility. Volatility then tends to relax to a stable level in the absence of any stimulating voltage input (or change in ambient conditions such as temperature).

In the examples below, we detail experimental evidence that further justifies and supports the theoretical motivation for using a framework involving volatility as a state variable.

JOULE HEATING In the case of filamentary switching behaviour in metal oxide memristors, the drift of oxygen vacancies is often nonlinear (exponential) with respect to the applied bias. This is attributed to multiple possible phenomenon, including high-magnitude local electric fields, and possible acceleration through Joule heating effects within the device [Abbey et al. \(2022\)](#). It is therefore the case that Joule heating may produce memory-dependent switching behaviour, with the internal temperature of the device - itself affected by the history of voltage applied across the device - affecting the rate of switching.

When the device is being programmed, energy is dissipated and held as heat energy in the device. Over time, if there is no stimulating voltage, Newton's law of cooling applies and the device returns to the ambient temperature. This is a dynamic process, and the cooling may not be fully complete before subsequent programming pulses. Cooling also occurs during the application of programming pulses, with cooling happening more rapidly, the higher the internal temperature of the device.

The actual picture governing volatility may not be fully described through the consideration of Joule heating effects. Wang et al. have examined the properties of temperature and diffusive dynamics for diffusion-driven, volatile tantalum-oxide memristors [Wang et al. \(2017\)](#). They state that temperature cannot be the driving factor in state relaxation for these diffusive, uni-polar devices, since temperature increases produced by Joule heating relax on the order of $1/\kappa$, whereas the memristor relaxation (stochasticity) occurs on a much longer order of $10 - 201/\kappa$. Thus, they hypothesise, diffusion is the main driving factor, rather than a remnant increase in temperature. This is applicable to the devices studied, which are based on the formation of conductive bridges formed from silver metal nanoparticles through SiO_xN_y dielectric substrate doped with silver ions, and sandwiched between Ag electrodes. Their conclusion is that Joule heating's role in the switching process, or subsequent relaxation (switching in the absence of bias) can vary dependent on the inherent volatility of the device.

STRUCTURAL RELAXATION IN PCM Structural relaxation is a phenomenon in [PCM](#) (e.g. Germanium Antimony Tellurium ([GST](#))), that results from the gradual consumption of point defects in the amorphous structure over time, following heating followed by rapid cooling that results in a glass-like structure, as introduced in Section 2.2 of Chapter 2.

The number of unconsumed defects in the device can be thought of as the level of volatility, which is highest after programming when the device has not had time to partially relax back into a crystalline configuration. As the defects are consumed, the volatility decreases. A more detailed description is as follows: after programming, the device relaxes towards the ideal glass configuration, through the consumption of defects in the amorphous structure

that cause the amorphous phase to differ significantly from the crystalline phase. Although drift does not cause the phase to become crystalline, it is generally agreed that defects in the amorphous phase are consumed, making the amorphous phase more similar in its chemical characteristics to the crystalline phase. Note that we emphasise that the structure is similar to the crystalline phase, but it is not identical. True relaxation to the crystalline phase would cause full restoration of the high resistive state [Gallo and Sebastian \(2020\)](#), which is possible if the device is not programmed fully to the amorphous phase using partial SET operations [Mantegazza et al. \(2010\)](#).

ELECTROCHEMICAL METALLIZATION: ENTROPY AND DISRUPTION OF STRUCTURE In [ECM](#) devices, the entropy of the oxygen vacancies in the device may also prompt a memory-dependent response. Minimisation of interfacial energy may prompt reconfiguration of metal ions within the insulator bulk for diffusive memristors. This phenomenon is described for the case of diffusive memristors in [Wang et al. \(2017\)](#).

The general form of volatility is a state variable which directly modifies the global rate of switching, whose increase is a function of external inputs such as voltage, or current, and which has a characteristic decay time constant, τ_χ . Example implementations of volatility are given in Section [3.4.4.3](#), where the effects of both Joule heating/Newton cooling, and of arbitrary volatile effects are considered.

3.2.6 Practical Simulations

Let us consider the switching of a single metastable switch to be modelled by a memoryless exponential distribution with the rate parameter λ proportional to the probability of switching. Note that this model is memoryless - the distribution for the evolution of the resistive state can be completely characterised by the current state, with no reference to the previous states of the device.

3.2.6.1 Point Process Sampling

Considering the cumulative distribution function for this exponential distribution, we can use inverse transform sampling (the Smirnov transformation) to generate samples for the point process:

$$\begin{aligned} p(T \leq t) &= 1 - e^{-\lambda t} \\ y &= 1 - e^{-\lambda t} \\ t &= -\frac{\ln(1 - y)}{\lambda} \end{aligned}$$

We can perform the simulation of parallel point process sampling in two ways. We note, building on the derivation in Section [3.3.1](#), that we can sample the time from a point process with rate equal to the sum of all rates of the independent point processes, and subsequently label the switching events post-fact. Alternatively, we can sample times for a cumulative

“up”-switching process and a cumulative “down”-switching process and take the smallest inter-arrival time as the next event, and repeatedly resample.

3.2.6.2 Changes to the Distribution at Events

When an event occurs (either a switching up or switching down event), this impacts the distributions and thus we need to reevaluate a sample for the switching time (for both the upwards and downwards processes).

We use the memoryless property of the exponential distribution, which states that if an event has not occurred by time τ then the distribution for an event to occur at time $t + \tau$ is still exponentially distributed. The only distinction is that we now modify the rate parameter for each distribution, which can be calculated as a simple linear scaling of the rate parameter, since the sum of N independent and identically distributed exponential distributions with rate parameter λ is an exponential distribution with rate parameter $N\lambda$, as described in Section 3.2.4.

3.2.6.3 Simulation Algorithm

The procedure for running the model simulation, for up and down switching, is given in Algorithm 1. R corresponds to the array of generated resistances (with r_{init} being the initial resistance, determined by the initial state parameter n) and T corresponds to the array of times at which the resistance changes. $\sim$ denotes sampling from a given probability distribution. f_{ro} is the readout equation that gives us a resistance value for a given state, as detailed in Equation 3.16.

Algorithm 1 Procedure for simulating the switching events of the event-based model.

```

1:  $R \leftarrow R_{\text{init}}$ 
2:  $n \leftarrow n(R)$ 
3:  $t \leftarrow 0$ 
4: while  $t \leq \Delta T$  do
5:    $t_{\text{down}} \sim \text{Exp}((N - n) \cdot \lambda'_{\text{down}})$ 
6:    $t_{\text{up}} \sim \text{Exp}(n \cdot \lambda'_{\text{up}})$ 
7:   if  $(t_{\text{down}} \leq t_{\text{up}} \text{ or } n = N)$  and  $n \neq 0$  then
8:      $n \leftarrow n - 1$ 
9:      $t \leftarrow t + t_{\text{down}}$ 
10:  else if  $t_{\text{up}} < t_{\text{down}}$  or  $n = 0$  then
11:     $n \leftarrow n + 1$ 
12:     $t \leftarrow t + t_{\text{up}}$ 
13:  end if
14:   $R \leftarrow R(n)$ 
15: end while

```

Note that in the case the value of n falls to zero, or reaches N , we simply wait for an appropriate event to occur, to increase or decrease the value of n , respectively.

3.2.7 Compatibility with Non-Event Based Modelling

3.2.7.1 Non-Event-Based Inputs

In order to incorporate non-event based inputs, including variables that may be continuous functions of time such as temperature or applied bias, or piecewise constant functions, we can sample these using threshold-based, or event-driven, sampling. For example, the volatility state variable, introduced in Section 3.2.5, is a continuous function of time. Given an independent input signal $m_i(t)$, we can define the associated marked point process, defined by the set of times $\mathcal{T}_i$, at which the events occur, to be:

$$\mathcal{T}_i := \{t : f_i(t) \in \mathcal{M}_i\}$$

where, for continuous functions the set $\mathcal{M}_i$ contains the function values, changing in steps of α :

$$\begin{aligned} \mathcal{M}_i = & \{f(0), f(0) + \alpha, \dots, \lim_{k \rightarrow \infty} f(0) + k\alpha\} \\ & \cup \{f(0), f(0) - \alpha, \dots, \lim_{k \rightarrow \infty} f(0) - k\alpha\} \end{aligned}$$

In contrast, for piecewise constant (discontinuous) functions, $\mathcal{M}_i$ will simply be the set of associated amplitudes at times $\mathcal{T}_i$, known ahead of time.

Due to the memoryless nature of the Poisson process, incorporating such sampling into the model is straightforward - whenever $t \in \mathcal{T}_i$, for a particular associated set of events i , we enforce the event and then resample times to firing for the Poisson switching processes. The canonical example of an input that can be modelled in this way, is the driving input voltage, that takes on predefined values at predetermined points in time.

Although the method of conversion for continuous functions for inclusion in the event-based modelling framework works to minimise the error associated with changes to parameters affecting the rate equations not matching those used during the simulation - which are dependent purely on external stimuli - additional challenges are presented in the case of variables that are instead, or also, functions of the device state themselves. For example, if Joule heating is dependent on the current through the memristor, then this variable must be re-evaluated whenever there is a change in the resistance, or in the voltage. As such, for variables in the rate equations that are functions of both external stimuli and the internal state variable, these should be re-evaluated every time any event happens, so as to avoid errors associated with the drift of the independent variables of the function away from the values used for the calculation.

We can incorporate these variables by considering them to be a subset of the variables by allowing the updates to be governed by function evaluation at pre-defined times, $\mathcal{T}_i$, as described previously, in order to account for external (autonomous) changes to the variable, in addition to at any point when a dependent variable of the function changes. Let $\{j_0 \dots j_j\} \in \{m_0 \dots m_i\}$ be the set of functions which depend on at least one other variable. We note that the

up and down switching rates themselves, $\lambda_{\text{up}}(n)$ and $\lambda_{\text{down}}(n)$, are within the subset $\{j_0 \cdots j_j\}$, since they are updated whenever the variable n is modified.

The incorporation of additional updates to parameters which are dependent on external stimuli, not limited to state switching events, is given in Algorithm 2.

3.2.7.2 Conversion to Discrete Time

If necessary, we can convert non-uniform series from the event-based model to uniform data in time through sampling, albeit introducing errors associated with the discretisation in time, as shown in Algorithm 3. $\mathcal{T}_{\text{switch}}$ denotes the ordered set of switching times, and $\mathcal{R}$ denotes the associated ordered set of values of a state variable at the times $\mathcal{T}_{\text{switch}}$. T_{sample} denotes the sampling period and T_{tot} denotes the maximum desired time to sample for.

3.3 MODEL ANALYSIS

Our model has thus far been defined, but we would also like a computationally efficient way of evaluating it. Our model is a marked⁴ Hawkes⁵ process with both self-excitatory and self-inhibitory behaviour. We would like to demonstrate different ways of looking at the behaviour of the model, in order to motivate the development of efficient schemes for numerical simulations and prediction of the statistics.

The majority of our analysis in this section will be under the assumption of static environmental conditions (temperature and bias), since under varying conditions, additional modifications to the switching parameters, and thus, to the equilibrium point of the model, will occur. The change of the memristive state in the absence of a voltage bias is usually referred to as the phenomenon of resistive drift, or volatility. The equilibrium point of the model under steady state conditions is identified in Section 3.3.3.

3.3.1 Equivalence to Other Point Processes

We can demonstrate that the parallel switching of N metastable switches, that can each take on one of two states, is equivalent to the simulation of a combination of two independent Poisson processes, one for the increase of a summative state variable, and one for the decrease of a summative state variable, happening independently. We can also show that these two Poisson processes can be thought of as a single marked Hawkes process with rate parameter $\lambda_{\text{tot}}(t)$, where each event changes the global rate up or down by a set amount, dependent on the current rate and the nature of the event (up or down).

We consider the scenario of a number of metastable switches at each time instant. This can be used to model either an interfacial or a filamentary type switching process. In the case of the interfacial switching process, we can consider each ion contributing to the conductivity of

⁴ A point process whose events are associated with additional information, such as amplitude. In this case - the direction of switching of the state variable.

⁵ A point process whose events impact the rate parameter (traditionally self-excitatory, but also allowing for self-inhibitory behaviour).

Algorithm 2 Event-based modelling procedure, incorporating additional non-event-based inputs.

```

1:  $R \leftarrow R_{\text{init}}$ 
2:  $n \leftarrow n(R)$ 
3:  $M_i \leftarrow \mathcal{M}_i[0]$ 
4:  $\lambda'_{\text{up}} \leftarrow f_{\text{up}}(\{M_i\})$ 
5:  $\lambda'_{\text{down}} \leftarrow f_{\text{down}}(\{M_i\})$ 
6:  $c_i \leftarrow 0$ 
7:  $t \leftarrow 0$ 
8:  $F \leftarrow 0$  ▷ State variable which denotes whether a firing event has occurred.
9: while  $t \leq \Delta T$  do
10:    $t_{\text{down}} \sim \text{Exp}((N - n) \cdot \lambda'_{\text{down}})$ 
11:    $t_{\text{up}} \sim \text{Exp}(n \cdot \lambda'_{\text{up}})$ 
12:   for all  $i$  do ▷ Firing for the state-independent variables
13:     if  $F = 0$  and  $c_i \leq |\mathcal{T}_i|$  and  $\mathcal{T}_i[c_i + 1] \leq \{t_{\text{up}}, t_{\text{down}}, \mathcal{T}_k[c_k + 1] \mid \forall k \neq i\}$  then
14:        $c_i \leftarrow c_i + 1$ 
15:        $t \leftarrow \mathcal{T}_i[c_i]$ 
16:        $M_i \leftarrow m_i(M_i[c_i], n, M_0, \dots, M_i, J_0, \dots, J_j)$ 
17:        $\lambda'_{\text{up}} \leftarrow f_{\text{up}}(n, M_0, \dots, M_i, J_0, \dots, J_j)$ 
18:        $\lambda'_{\text{down}} \leftarrow f_{\text{down}}(n, M_0, \dots, M_i, J_0, \dots, J_j)$ 
19:        $F \leftarrow 1$ 
20:     end if
21:   end for
22:   if  $F = 0$  and  $(t_{\text{down}} \leq t_{\text{up}} \text{ or } n = N)$  and  $n \neq 0$  then
23:      $n \leftarrow n - 1$ 
24:      $t \leftarrow t + t_{\text{down}}$ 
25:     for all  $j$  do ▷ Update the state-dependent variables
26:        $J_i \leftarrow j_i(n, M_0, \dots, M_i, J_0, \dots, J_j)$ 
27:        $\lambda'_{\text{up}} \leftarrow f_{\text{up}}(n, M_0, \dots, M_i, J_0, \dots, J_j)$ 
28:        $\lambda'_{\text{down}} \leftarrow f_{\text{down}}(n, M_0, \dots, M_i, J_0, \dots, J_j)$ 
29:     end for
30:   else if  $F = 0$  and  $(t_{\text{up}} < t_{\text{down}} \text{ or } n = 0)$  then
31:      $n \leftarrow n + 1$ 
32:      $t \leftarrow t + t_{\text{up}}$ 
33:     for all  $j$  do
34:        $J_i \leftarrow j_i(n, M_0, \dots, M_i, J_0, \dots, J_j)$ 
35:        $\lambda'_{\text{up}} \leftarrow f_{\text{up}}(n, M_0, \dots, M_i, J_0, \dots, J_j)$ 
36:        $\lambda'_{\text{down}} \leftarrow f_{\text{down}}(n, M_0, \dots, M_i, J_0, \dots, J_j)$ 
37:     end for
38:   end if
39:    $R \leftarrow R(n)$ 
40:    $F \leftarrow 0$ 
41: end while

```

Algorithm 3 Procedure for conversion of the non-uniform time series to uniformly sampled time series.

```

1:  $i \leftarrow -1$ 
2:  $t_{\text{curr}} \leftarrow 0$ 
3:  $\mathcal{R}_{\text{sample}} \leftarrow \{\}$ 
4: for all  $t \in \mathcal{T}_{\text{switch}}$  do
5:   if  $t \geq T_{\text{curr}}$  then
6:     for all  $n \in [1, 1 + \lfloor \frac{t - t_{\text{curr}}}{T_{\text{sample}}} \rfloor]$  do
7:        $\mathcal{R}_{\text{sample}} \leftarrow \mathcal{R}_{\text{sample}} \cup \{\mathcal{R}[\max(0, i)]\}$ 
8:        $t_{\text{curr}} \leftarrow t_{\text{curr}} + T_{\text{sample}}$ 
9:     end for
10:     $i \leftarrow i + 1$ 
11:    if  $t_{\text{curr}} > T_{\text{tot}}$  then break
12:    end if
13:  else
14:     $i \leftarrow i + 1$ 
15:  end if
16: end for
17: if  $\lfloor \frac{T_{\text{tot}} - t_{\text{curr}}}{T_{\text{sample}}} \rfloor > 0$  then
18:  for all  $n \in [1, \lfloor \frac{T_{\text{tot}} - t_{\text{curr}}}{T_{\text{sample}}} \rfloor]$  do
19:     $\mathcal{R}_{\text{sample}} \leftarrow \mathcal{R}_{\text{sample}} \cup \{\mathcal{R}[\max(0, i)]\}$ 
20:  end for
21: end if
22: return  $\mathcal{R}_{\text{sample}}$ 

```

the insulating channel to be a single switch, with the values of all the switches being summed in parallel.

Note that this kind of modelling can be used to model a wide variety of processes. The only distinction will be the readout equation which gives us the instantaneous effective resistance based on the state (the I-V relation). There are no special requirements about the nature of the function other than it be a function (i.e., it is not one-to-many, which would indicate the presence of other unconsidered state variables).

Considerations must be made for the model of conduction and of switching in calculating the readout equation from the state variable (i.e., from the number, or proportion, of switches in the on state). In the case of an interfacial type switching behaviour or a filamentary model for the memristance, we might model the extent/length of the conductive filament or interfacial barrier through the device as being dependent on the proportion of switches in the high state. In the case of the area-dependent switching of an interfacial barrier, we might model the extent as being proportional the root of the number of switches in the high state. The other consideration is modelling of the conduction process. In scenarios where we model the conduction process as mainly Ohmic, in the case of an interfacial barrier model, we might, for example, have the readout equation proportional to the reciprocal of the total number of switches in the on state, modelling the switches as parallel conductive paths. Otherwise, we might consider the number of switches to be proportional to the length of a conductive filament for an area-independent model, resulting in direction proportionality between the

resistance readout and the state variable. In the case of the Schottky barrier conduction model, the dependence of the resistance on the number of switches might be highly nonlinear.

Consider two independent Poisson processes, P_1 and P_2 , with a rate parameters λ_1 and λ_2 respectively. Here, the inter-arrival time distribution T_1 of P_1 describes the transition of a switch from a low state to a high state, while inter-arrival time distribution T_2 of process P_2 , describes the transition of a switch to a high state from a low state.

We now consider the distribution of a random variable that is defined as the minimum time of the interarrival time distributions. The density of this process, which we will model as the random variable S , will also be an exponential distribution, with a rate parameter equal to the sum of the individual rates of the two processes.

The ratio of the frequency of the events of type P_1 to those of type P_2 will be such that events of type P_1 happen with frequency λ_1 and events of type P_2 happen with frequency λ_2), as can be seen from the probabilities $p(T_1 > T_2)$ and $p(T_2 > T_1)$:

$$\begin{aligned}
 p(T_1 > T_2) &= \int_0^\infty \int_0^{t_1} \lambda_1 e^{-\lambda_1 t_1} \lambda_2 e^{-\lambda_2 t_2} dt_1 dt_2 \\
 &= \int_0^\infty \left[\lambda_1 e^{-\lambda_1 t_1} \frac{\lambda_2 e^{-\lambda_2 t_2}}{-\lambda_2} \right] dt_2 \\
 &= \int_0^\infty -\lambda_1 e^{-\lambda_1 t_1} e^{-\lambda_2 t_1} + \lambda_1 e^{-\lambda_1 t_1} dt_2 \\
 &= \left[-e^{-\lambda_1 t_1} + \frac{\lambda_1}{(\lambda_1 + \lambda_2)} e^{-\lambda_1 t_1} e^{-\lambda_2 t_2} \right]_0^\infty \\
 &= 1 - \frac{\lambda_1}{(\lambda_1 + \lambda_2)} = \frac{\lambda_2}{(\lambda_1 + \lambda_2)}
 \end{aligned}$$

and by symmetry, we see that:

$$p(T_2 > T_1) = \frac{\lambda_1}{(\lambda_1 + \lambda_2)}$$

The significance of this observation is that, since a Poisson process is memoryless (the occurrence of an event has no bearing on the occurrence of future events), we can equivalently simulate the simultaneous switching of many switches by calculating the time to the first event that occurs, recording the event, and then running a new simulation. This argument is valid for more than two processes happening simultaneously. This is straightforward to show inductively by considering the addition of a new parallel process P_3 and considering the ratio of events of type P_3 to those of type P_1 or P_2 , and so on.

3.3.2 Distribution after Delay

The distribution, and statistics such as the mean and variance, of the state of the random process after a delay ΔT are not straightforward to calculate, given the fact that λ_{up} and λ_{down}

change following any event, as well as due to the indeterminate number of events in a given timeframe.

However, under certain conditions, an assumption can be made that the total rate of events is relatively constant (for a marked process which is the sum of the up and down switching processes). This is always true if $\lambda'_{\text{down}} = \lambda'_{\text{up}}$, and is approximately true if $\lambda'_{\text{down}} \approx \lambda'_{\text{up}}$. This can be seen from the calculation of the rate for a combined, marked process as

$$\lambda_{\text{tot}} = (m) \cdot \lambda'_{\text{down}} + (N - m) \cdot \lambda'_{\text{up}} \quad (3.7)$$

If $\lambda'_{\text{down}} \neq \lambda'_{\text{up}}$, the error resulting from this assumption will be dependent on the deviation of the rate over time from the initial rate as a result of switching events.

In order to efficiently calculate the mean change in the state over a specific time period for a given initial state, we can make constant rate assumptions for both the up and down switching processes (and thus the relative probabilities of up and down switching) and simply find the estimated mean change in the state, $\hat{\mu}$, as:

$$\text{Let } \begin{cases} a = \lambda_{\text{up}} \Delta T \\ b = \lambda_{\text{down}} \Delta T \end{cases}$$

$$\hat{\mu} = \mathbf{E}[N(\Delta T)] \frac{a - b}{a + b}$$

If we further relax our approximation and assume that the total rate is constant, but allow the relative probabilities of the up or down switching to vary, we can obtain a more refined approximation of the distribution over the final states after a time ΔT , $\mathbf{S}(\Delta T)$. The mean final state and the variance in the final state, can be determined by a polynomial of the Markov transition matrix, M , as shown in Equation 3.8 and the initial state vector L . The random variable $Q(\Delta T)$ represents the count process for the initial combined Poisson point process with rate parameter $\lambda_{\text{tot}} = \lambda_{\text{up}} + \lambda_{\text{down}}$ (with m fixed at its initial value at $t = 0$):

$$\mathbf{S}(\Delta T) = \sum_{i=1}^{\infty} P(Q(\Delta T) = i) M^i L \quad (3.8)$$

Intuitively, this is a marginalisation over the number of events, i , that occur in the time leading up to ΔT , of the expected value of the state vector given that number of events have occurred.

Note that the special structure of our point processes dictates that the state transition matrix, $M \in \mathbf{R}^{N \times N}$, has zeros everywhere apart from indices where $j = i + 1$ (the superdiagonal) or $j = i - 1$ (the subdiagonal) to represent up and down transitions, respectively. The values are given as follows:

$$\text{Let } \begin{cases} a = \lambda'_{\text{up}}(i) \\ b = \lambda'_{\text{down}}(N - i) \end{cases}$$

$$M[i, j] = \begin{cases} \frac{a}{a+b} & \text{if } j = i + 1 \\ \frac{b}{a+b} & \text{if } j = i - 1 \\ 0 & \text{otherwise} \end{cases} \quad (3.9)$$

The expected value and the variance can then be determined from the final distribution vector $S(\Delta T)$ as shown in Equations 3.10 and 3.11, where $R(n)$ is the readout equation, mapping the state variable to the resistance values.

$$\mu_{S(\Delta T)} = \sum_{i=1}^N S(\Delta T)[i] R(i) \quad (3.10)$$

$$\sigma_{S(\Delta T)}^2 = \sum_{i=1}^N S(\Delta T)[i] (R(i) - \mu_{S(\Delta T)})^2 \quad (3.11)$$

3.3.3 Calculation of the Equilibrium Point (Metastable Point)

Given the forward switching rate, we can calculate the equilibrium number of switches in the low resistance state at a given time by equating λ_{down} to λ_{up} . The expression for this is given in Equation 3.12, below. A point of dynamic equilibrium is attained when the rate $\lambda_{\text{down}} = \lambda_{\text{up}}$; that is, the rate of switching from the high resistance state to the low resistance state is equal to the rate of switching in the opposite direction. In the case that we have a number of states greater than 2, the determination of the equilibrium state is less straightforward and the steady state vector of the associated Markov state transition matrix must be identified.

$$\lambda_{\text{up}} = \lambda_{\text{down}}$$

$$n_{\text{eq}} e^{-\frac{E_a - V(t)q/2 - E_{\text{off}}/2}{k_b T(t)(1+\rho(t))}} = m_{\text{eq}} e^{-\frac{E_a + V(t)q/2 + E_{\text{off}}/2}{k_b T(t)(1+\rho(t))}}$$

$$n_{\text{eq}} e^{\frac{2(V(t)q/2 + E_{\text{off}}/2)}{k_b T(t)(1+\rho(t))}} = m_{\text{eq}}$$

$$n_{\text{eq}} = \frac{N}{e^{\frac{(V(t)q + E_{\text{off}})}{k_b T(t)(1+\rho(t))}} + 1} \quad (3.12)$$

We can see that the equilibrium point, as a fraction of N , is independent of the value of E_a , depending only on E_{off} (as well as $T(t)$, $V(t)$, and $\rho(t)$). This shows that the equilibrium point and the rate of attainment of the equilibrium point can be independently adjusted, using E_{off} and E_a , respectively.

The equilibrium resistance can be determined from the overall equilibrium state via the readout equation.

When external environmental parameters are functions of time, the model's equilibrium point and the rate parameters associated with the switching point processes will also be functions of time, changing even when there are no underlying state changes due to metastable switching events.

Under a zero- or a piecewise constant- bias setting, the phenomenon known as resistive drift occurs for memristors. This can be shown to be a competition between two Poisson processes with linearly varying rates. There is a well-defined equilibrium point towards which resistive drift occurs, which can be calculated by finding the state for which the rates of the competing switching processes are equal. In the steady state, in a constant-bias setting, the temperature will no longer be time varying. In a zero-bias setting specifically, the temperature will be equal to the ambient temperature. The zero-bias equilibrium point therefore has a simplified expression given as follows:

$$n_{eq} = \frac{N}{e^{\frac{qV_{off}}{k_B T}} + 1} \quad (3.13)$$

$$(3.14)$$

We can also express the offset voltage V_{off} in terms of n_{eq} as:

$$V_{off} = \ln \left(\frac{N - n_{eq}}{n_{eq}} \right) \cdot \frac{k_B T}{q} \equiv \ln \left(\frac{m_{eq}}{n_{eq}} \right) \cdot V_T \quad (3.15)$$

where $V_T = \frac{k_B T}{q}$ is a constant, often referred to as the thermal voltage.

3.3.4 Additional Discussion

Here, we detail a number of additional considerations regarding the model, and look at the strengths and potential weaknesses of the modelling approach.

3.3.4.1 Coarse and Fine Simulations

One strength of a discrete modelling strategy, with a finite number of possible system states (in our case, N), is that simulations can be conducted at different resolutions, depending on precision requirements, and the scale of interest. For example, if a memristive device is to be used as a binary storage device, where only the states 1 and 0 are distinct, then a value of $N = 2$ suffices to model the readout value of interest. In contrast, for a simulation where multiple stable states are to be for information storage, a larger value of N may be required. The “best” value of N to use is a matter of perspective - it depends on the extent to which regions of device resistance or behaviour are considered to be equivalent for the intended purpose, as

exemplified by the binary storage example, where the many possible configurations of the device are summarised by a single bit (2 possible values) of information.

3.3.4.2 Error Compared to Timestep-Based Method

Note that the error, in contrast to the [GMSM](#), is related to the level of quantisation and the type of quantisation. In the case of the previous method, we have errors related to missing switching events, which impact the instantaneous probability distribution. We also have error associated with the approximation of a sum of binomials by a normal distribution. The normal distribution's parameters at each point in time will change with each switching event; however, this is not reflected by the choice of the use of a constant delay between each distribution update. The duration of the delay chosen will have an impact on the accuracy as well.

In contrast, in the case of event-based switching, no error is introduced if the parameter N accurately reflects the physical mechanism. Using a value of N that is smaller will introduce a quantisation error (but no bias) to the switching process evaluation. That is - the expected value of the process after given number of switching events will be the same, since the value of n_{eq} scales linearly with N , and the associated parameters of the readout equation can be adjusted accordingly.

Note that integrate and fire sampling for continuous functions of time, as described in Section [3.2.7.1](#), will unavoidably introduce discretisation errors. If the functions vary slowly in time, then the number of additional events due to the introduction of additional functions will be small. However, if the functions vary quickly in time and the number of generated events is large, then the benefits of event-based modelling approaches diminish, and the model's events become dense in time. This still, however, offers benefits over a uniform sampling approach, since it associates the density of samples in time with the rate of information change that impact the model updates (and hence the error associated with discretisation in time).

3.3.4.3 Impact of the Parameter N

As a consequence of the fact that the variance and mean of a Poisson process are equal to the rate of the process, the magnitude of the parameter N will scale the expected value and the variance of the final state in a linear fashion. As a result, the standard deviation in the final state, after a given delay, is proportional to the square root of N . Thus, we can achieve less stochastic switching by scaling up N . Achieving the same mean, for a given delay, for the readout variable can be achieved by applying a straightforward linear scaling of the parameter n in the readout equation. When we increase N , the average jump time decreases linearly with N (the rates of up and down jumps increase proportional to N). This means that over the timescale of interest, many more trials occur, with each trial also contributing to a smaller percentage change in the resistance, resulting in a decrease in the variance with increasing N . Continuous and deterministic switching behaviour can be approximated in the limit as $N \rightarrow \infty$.

This same principle underlies decreasing variance with increasing number of trials in any kind of statistical estimate using independent trials. It is a general rule that macroscale

behaviour (averaged over many independent trials) is always more predictable in physical systems, due to averaging of individual stochastic phenomena.

In the case of the normal approximation of the [GMSM](#), the total number of metastable switches also has an impact on the variance of the stochastic steps, but not on their mean (expressed in terms of a change of resistance, which is a parameter dependent on the ratio of n_A to N). This is, similarly, because for each of the normal approximations, an increase in the number of trials leads to a linearly proportional increase in the number of expected successes, and also a linear increase in the variance. When these normal approximations are divided by the number of trials to normalise, the mean is divided by N , but the variance is divided by N^2 . This results in the mean being independent of N , while the variance scales linearly with $1/N$.

3.4 RRAM METAL OXIDE DEVICE EXAMPLE

In this section we look at an example of the modelling framework applied to a titanium dioxide memristive device, to model the resistive drift phenomenon under zero bias.

3.4.1 *Physical Model*

Modelling of metal oxide [MIM](#), and more specifically - Titanium Dioxide [MIM](#), devices has been studied previously. Titanium Dioxide memristor switching has been previously modelled as a single conductive pathway of varying resistance [Strukov et al. \(2008\)](#), and other similar devices, such as tantalum oxide memristors have been modelled as exhibiting the dominance of a single filament or conductive region [Strachan et al. \(2013\)](#).

Based on initial investigation of a drift dataset (see Section [3.4.3](#) below), we model the conductance of the device as linearly proportional to state variable n . Based on this information, we hypothesise that the mechanism underlying the switching of the device modulates the width of a conductive region within the active substrate, bridging a connection between the electrodes. Phenomena that could account for this behaviour include a single conductive filament that grows or reduces in width, or multiple conductive filaments that each contribute linearly to the conductance of the device. We observe that it is necessary to use a value of n_{extra} and m_{extra} much greater than the value of N . This suggests a thresholding effect, where the rate of switching is relatively constant in the switching region, under a constant voltage bias. This corresponds somewhat to the ideal behaviour for a memristor, which is why such titanium dioxide devices were presented as candidate memristive devices originally [Strukov et al. \(2008\)](#). We assume a large $n_{\text{extra}} = 10 \times N$ such that the switching rates remain relatively constant across the considered range of resistance values.

Our model suggests that a large number of switches must be in the low resistance state before there is any measurable impact on the rate. Switching a sufficient number of the switches into the low resistance state can be thought to be performed during the electroforming step. We can imagine our model being a conductive filament that grows in length, eventually making physical contact with the electrode on the other side of the active layer, allowing electrical conduction to occur. Once this contact occurs, additional switching causes the contact region

to grow linearly in area of contact, resulting in a linear increase in the conductive state once we are in this “active” region. The model for this process is visualised in Figure 3.6.

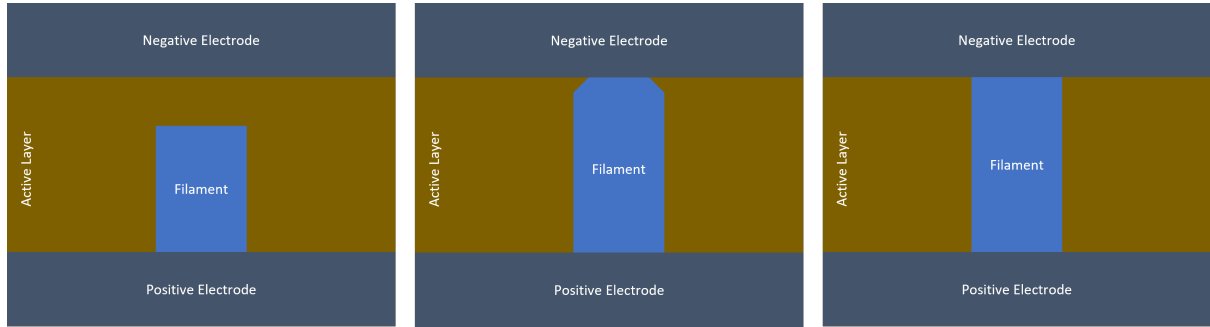


Figure 3.6: Our physical model for switching in a titanium dioxide memristor. The growth of the filament will not significantly impact the resistance (conductance) - i.e. the readout value - until the filament size crosses a particular threshold. Once this condition is achieved (middle figure), we are in the active region of switching, where small changes in the state variable have a significant impact on the resistance, due to the modulation of an effective area of conduction through the active layer of the device.

3.4.2 Readout Equation

In the case of Titanium Dioxide devices, we assume that the state variable constitutes the number of parallel filaments that have been formed through the device, or alternatively, the diameter of a single conductive filament. In both cases, the resistance will not be linearly related to this state variable. Rather, it is the conductance, thus our readout equation will be:

$$R(n) = \frac{1}{g_{\text{diff}} \max(n, n_{\text{thresh}}) + \frac{1}{R_{\text{high}}}} \quad (3.16)$$

We can see from the equation that the behaviour of the model is thresholded, with the resistance remaining at the value R_{high} until the value of n surpasses the parameter $0 \leq n_{\text{thresh}} \leq N$, upon which the conductance increases linearly in steps of g_{diff} . This is motivated by the conductive filament model described above.

We can also invert the equation to find R in terms of n :

$$n(R) = n_{\text{thresh}} + \frac{1}{g_{\text{step}}} \left(\frac{1}{R} - \frac{1}{g_{\text{parallel}}} \right) \quad (3.17)$$

This equation is useful for determining the correspondence between a certain resistance value and the number of switches in the “low” state, n , for example, for determining the desired n value for a specified equilibrium resistance. Note that if the resistance is at its maximum, we simply assume the value of n to be n_{thresh} , although it could be any value at or lower than this. This is based on the assumption that we lie in or close to the active switching region once the device has been electroformed.

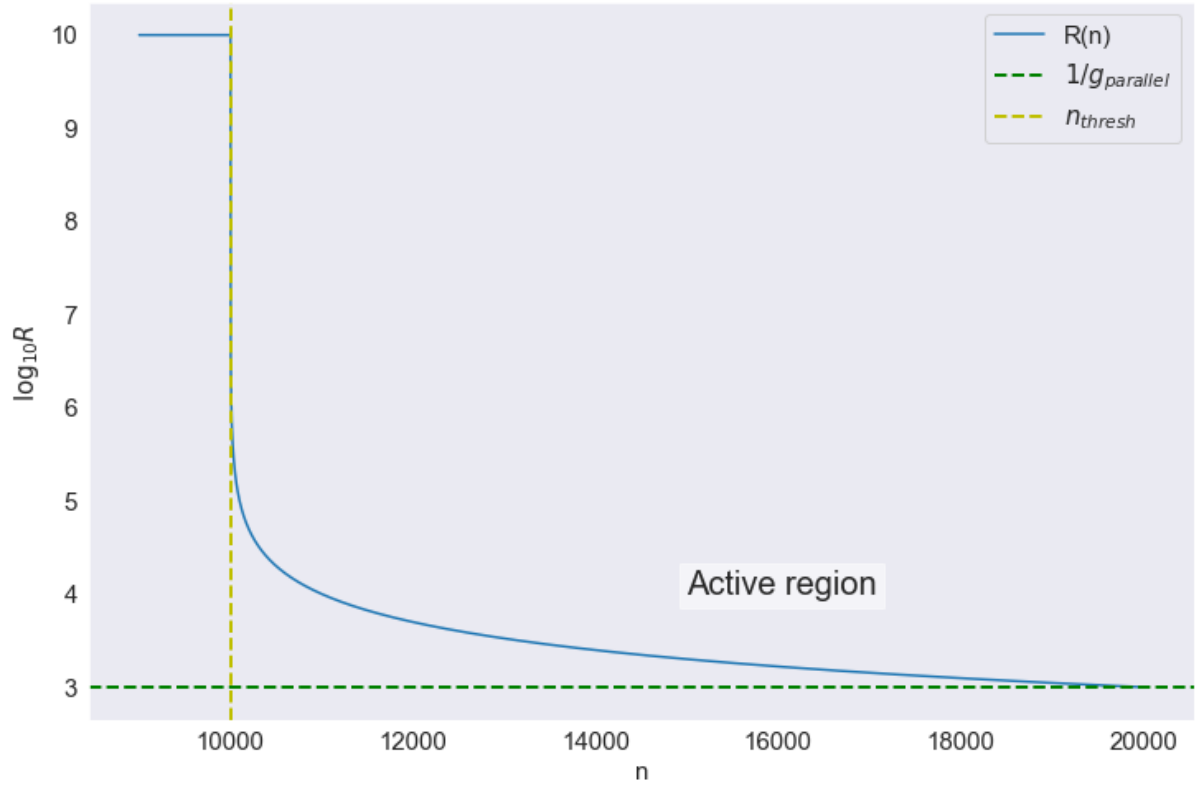


Figure 3.7: The active switching region is defined and shown as the region where changes to the state variable measurably impact the readout variable (in this case the resistance). Before this point, changes to the state are modelled not to have an impact.

Note that though the readout equation is not linear in the state variable, the equivalent conductance (reciprocal of the resistance) is linear in the state variable, the physical motivation for which is provided above. From the readout equation, we can see that the active region will begin at approximately $n = n_{\text{thresh}}$. This is visualised in Figure 3.7.

3.4.3 Dataset

In order to evaluate the behaviour of the model under controlled conditions, we created a dataset based on the evolution of memristive states [El-Geresy et al. \(2024a\)](#).

We collected 2751 device hours of data across two separate memristor arrays. This involved setting each of the devices in the array to a random initial state with a particular resistance and measuring the evolution of the resistance over time, using a short pulse of 0.3 Volts (V) magnitude to read the instantaneous resistance every 10 seconds. Table 3.1 shows the makeup of the dataset. In total, 208 data series were collected, of varying length. The devices used were thin film Titanium Dioxide (TiO_2 memristors, placed only on the diagonal of crossbar arrays, so as to prevent the influence of sneak path currents. The measurements were collected using the ArC ONE memristor characterisation platform [ArC Instruments LTD \(2019\)](#).

After collection, the series were inspected and a data cleaning step was performed. As part of this step, series for memristors which had failed or degenerated to a non-volatile state after programming, as well as clearly anomalous measurements within series due to measurement

Time (Hours)	Number of Devices	Array Number
9	10	1
5	7	1
11	7	1
5	9	1
12	9	1
5.5	8	1
9	8	1
9.5	8	1
9.33	8	1
12	9	1
14	9	1
11	9	1
11	8	1
33	16	2
12	14	2
15	15	2
17	11	2
13	13	2
12	9	2
36	9	2

Table 3.1: Drift dataset individual experiments. Each experiment involved one of the two arrays available. The memristors on the array were randomly programmed to values within the operating range, then periodically monitored for a number of hours (with one resistance reading taken every 10 seconds).

errors, were removed. Following data cleaning, 14 series were excluded from the dataset, leaving 196 series.

3.4.3.1 Quantisation

Figure 3.8 shows a histogram of the unquantised resistance values in the dataset, showing the range of interest to be the interval of approximately $[10^3, 10^5]$, where changes to the value of n significantly affect the output resistance. As such, we choose the values of N , n_{thresh} , g_{step} , and g_{parallel} accordingly. We set $g_{\text{step}} = 10^{-7}$, $g_{\text{parallel}} = 10^{-10}$, and $N = 2n_{\text{thresh}} = 20,000$.

After choosing the parameter values that determine the number of resistance states and their values, we calculate the resistance corresponding to each possible value of n , using the readout equation given in Equation 3.16. Let $R(n)$ be the readout equation mapping the state variable n to the corresponding resistance value. The resistance bin boundaries, $\mathcal{R}_{\text{bound}}$ and values $\mathcal{R}_{\text{val}}$, in the order of decreasing resistance (increasing state), are given by:

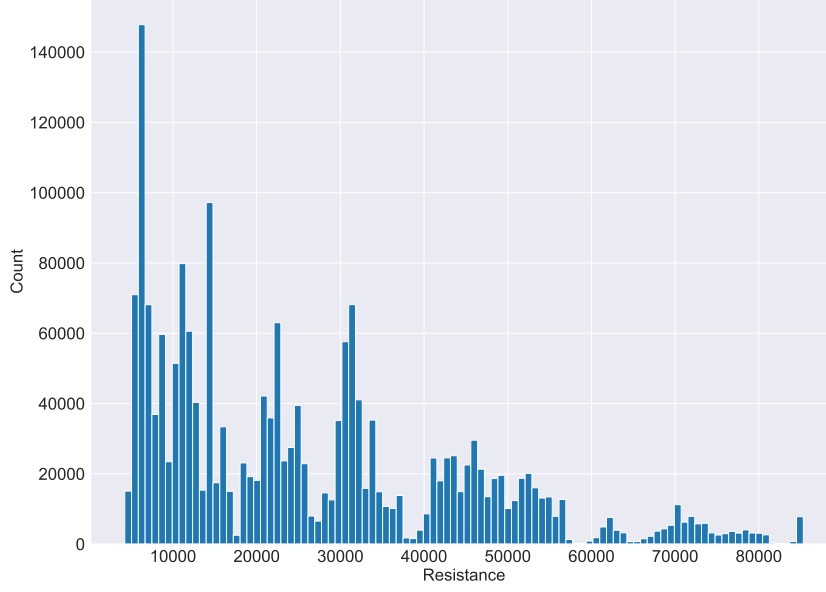


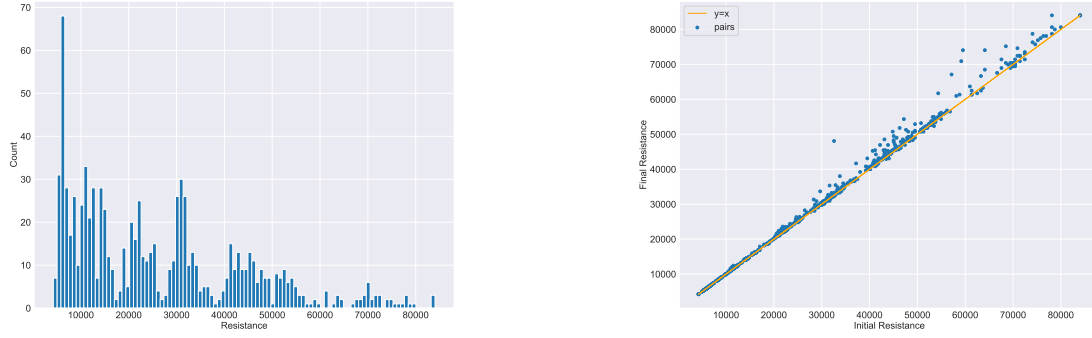
Figure 3.8: The unquantised resistance dataset, showing the empirical distribution of all resistance values in all series.

$$\begin{aligned}
\mathcal{R}_{\text{bound}} &= \{-\infty, R(n_{\text{thresh}} + 0.5), \\
&\quad R(n_{\text{thresh}} + 1.5), \dots, R(N - 0.5), \infty\} \\
\mathcal{R}_{\text{val}} &= \{R(n_{\text{thresh}}), R(n_{\text{thresh}} + 1), \\
&\quad \dots, R(N - 1), R(N)\} \\
\text{Where } \begin{cases} |\mathcal{R}_{\text{bound}}| &= N - n_{\text{thresh}} + 2 \\ |\mathcal{R}_{\text{val}}| &= N - n_{\text{thresh}} + 1 \end{cases}
\end{aligned}$$

3.4.3.2 Pairs

For each series of quantised resistance values in the dataset, we obtain pairs of resistance values from the sequential, non-overlapping intervals of length $T = 10,000$ seconds (2.778 hours). This yields a dataset consisting of pairs of initial and final resistance values for the given interval, allowing us to evaluate the statistics of the resistance drift for such an interval, conditioned on the initial resistance value.

A histogram of the initial values in the pairs of quantised resistance values for a value of $\Delta T = 10,000$ is shown in Figure 3.9a. A plot of all the pairs of values in the dataset is shown in Figure 3.9b. We can see a stochastic tendency for the device's resistance to increase over time.



(a) A histogram of the initial resistances in the dataset of generated pairs of initial and final resistances for $\Delta T = 10,000$ seconds.

(b) A scatter plot of all the pairs in the dataset generated for $\Delta T = 10000$ seconds. We see that there is a stochastic tendency for the device resistance to increase over time.

Figure 3.9: Figures showing the statistics of the generated pairs dataset.

3.4.4 Model Fitting

3.4.4.1 Linear Conductance Approximation

To model the impact of resistive drift, we measured the mean change in the conductance conditioned on each of the initial quantised resistance values, $\mathcal{R}_{\text{val}}$, we calculate the mean change in the resistance across all the pairs in the dataset for given initial resistances. A plot showing this is shown in Figure 3.10.

Based on the physical model described in Section 3.4.1, we use the approximation that the conductance changes linearly, at a rate proportional to the time, at a rate that is independent of the initial state. We had observed that this modelling approach appeared to fit a trend in the mean changes. The proposed linear conductance model of drift is given as follows, with parameter α - the change in the state variable - fit to the data:

$$\Delta R(R_{\text{init}}) = \frac{1}{g_{\text{parallel}} - \alpha g_{\text{diff}}} - R_{\text{init}} \quad (3.18)$$

We compare the goodness of fit of our model to two other models: the stretched exponential model (Equation 3.19), which has been previously cited as a good model of resistive drift in memristors [Abbey et al. \(2022\)](#), with τ , β , and γ as fitting parameters; and a linear resistance model (Equation 3.20), with m and c as fitting parameters.

$$\Delta R_{\text{se}}(R_{\text{init}}) = \gamma \cdot (1 - e^{-(\frac{\Delta T}{\tau})^\beta}) + R_{\text{init}} \cdot (e^{-(\frac{\Delta T}{\tau})^\beta} - 1) \quad (3.19)$$

$$\Delta R_{\text{lr}}(R_{\text{init}}) = m \cdot R_{\text{init}} + c - R_{\text{init}} \quad (3.20)$$

In order to evaluate the goodness of fit of our model and the stretched exponential model, we split our pairs dataset into two parts - training (80% of the datapoints) and test (20%). Based on this split, we fit each model to the data using an iterative trust region fitting algorithm [Branch et al. \(1999\)](#) to minimise the squared error associated with fitting the function to the predicted resistance change in the training data. We then compare the fitting error evaluated on the training data to that when the fitted model is run on the unseen test data.

We test two different experimental settings, using a separate approach for splitting the dataset into train and test data each time. The training and test errors for each of these settings are shown in Table 3.2. For the “Random” splitting approach, we randomly select 20% of the datapoints to use as test data. For the “Interpolation” approach, we choose the central 20% of points, ordered in terms of the initial resistance. We use the mean absolute error in the predicted change in resistance as a measure of the goodness of fit of the model.

Model	Splitting Approach			
	Random		Interpolation	
	Train Error	Test Error	Train Error	Test Error
Stretched Exponential	554.3	727.9	579.6	676.8
Linear Resistance	461.1	607.0	488.8	554.1
Linear Conductance	429.9	572.5	457.5	521.5

Table 3.2: Training and test errors for fitting the proposed linear conductance model, the linear resistance model, and the stretched exponential model to the pairs dataset, for two different dataset splits.

We see that the proposed linear conductance model outperforms model outperforms both the stretched exponential model and the linear resistance model, in terms of the train and test fitting errors.

Following our evaluation and comparison, we fit the model to the entire pairs dataset to obtain the best fit parameters. For the chosen timestep of 10,000 seconds, we find a value of $\alpha = 3.62$ yields the best fit, for the given parameters g_{diff} , g_{parallel} , N , and n_{thresh} . This associated line of best fit for this linear conductance change model is shown in Figure 3.10. The moving averages are plotted to enable a clearer view of the general trend. Although their shape is clear, they do include bias associated with the (symmetric) averaging window size, with a larger window size tending to overestimate the mean change, as can be seen from the larger magnitude of the moving average plot for a window size of 50, compared to the model fit to the data. The moving average window of size 5 yields a (stochastic) function whose mean closely matches the model.

3.4.4.2 Parameter Fitting and Constant Rate Approximation

We would like to obtain parameters that allow the mean behaviour of the model to correspond to the mean predictions of the linear conductance model. Due to the difficulty in obtaining an analytic expression for the delay-conditional distribution, as seen in Section 3.3.2, we use a constant rate approximation to obtain approximate mean values for this distribution over time. We assume that the up and down switching rates of the model remain relatively constant. In

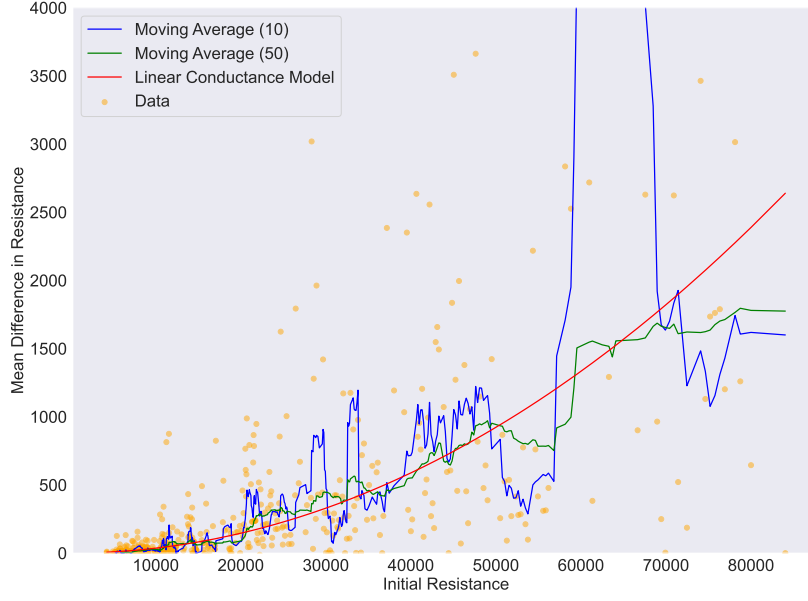


Figure 3.10: The mean change in resistance for the dataset of pairs with $\Delta T = 10000$. We show a moving average of the data with different averaging window sizes (5 and 50), which give a clearer picture of the nonlinear trend. The linear conductance model with best fit parameters is plotted alongside the data and trend lines.

order to fit the model, we calculate the rate at a value of $n = n_{\text{thresh}}$ and make the assumption that the rate remains equal to this rate over the region of interest. Thus, this approximation is only valid if the quantities $\frac{n_{\text{thresh}} - a}{n_{\text{thresh}}}$ and $\frac{N - n_{\text{thresh}} + a}{N - n_{\text{thresh}}}$, approximating the percentage change in the up and down switching rates remain relatively close to 1.

We choose the parameters N and n_{thresh} accordingly. As such, we require $n_{\text{thresh}} \gg a$ and $N - n_{\text{thresh}} \gg a$. We must also ensure that $N + 1 - n_{\text{thresh}}$, defining the number of unique resistance states according to Equation 3.16, is appropriately selected to give a sufficiently fine quantisation over the switching region for which we have data.

For the given value of the parameter a , we can determine an appropriate value of V_a . We first select a value of V_{off} to ensure an equilibrium point (see Section 3.3.3) that is far above the active switching region (in terms of resistance), to ensure that the assumption that the rate is constant over the considered range holds, since the equilibrium point at a particular bias sets the zero-crossing point for the rate equation, hence our constant rate assumption will break down as the value of n approaches n_{eq} . In our case, this corresponds to setting an equilibrium point with a value of $n \ll n_{\text{thresh}}$. We can see in Figure 3.11 how the choice of V_{off} impacts n_{eq} ; and hence, the constant rate assumption, and the fit to the linear conductance model. Assuming a constant rate and a given value of V_{off} , we can derive a value of V_a according to the following equation:

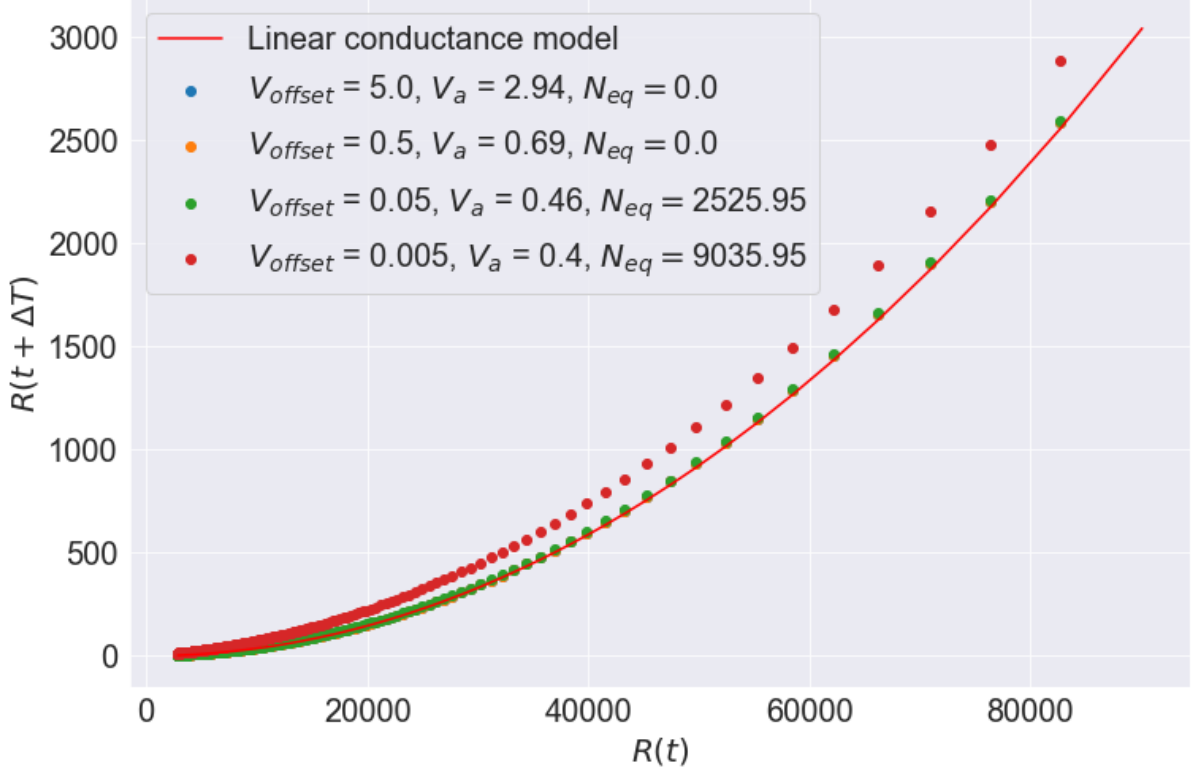


Figure 3.11: Fitting the event-based model to the linear conductance model. We use a constant rate assumption to derive approximate values for the parameters.

$$V_a = -\frac{k_B T}{q} \ln \left(2\Delta T n_{\text{thresh}} \sinh \left(V_{\text{off}} \frac{q}{2k_B T} \right) \right) \quad (3.21)$$

For very small values of V_{off} , the constant rate assumption will not hold, since the equilibrium point of the model will be high and closer to the active region.

In Figure 3.12, we see averaged zero-bias simulations from the event-based model, based on the fit parameters, showing the mean change in the resistance fitting to the linear conductance model, as well as the associated variance of the evolutions (which increases with the magnitude of the resistance).

3.4.4.3 Volatility

In the case of thin film memristors, volatility can be decomposed into two variables. The first of these is the temperature (that may increase due to Joule heating) and the second is a dimensionless volatility state variable to account for additional volatile effects, such as disruptions to the internal structure of the memristor on application of a bias voltage and subsequent restoration of order following its removal (interfacial entropy minimisation).

In modelling the effects of Joule heating, we can assume that a proportion of the current flowing through the device (proportional to the square of the current flowing through the device) is converted to heat energy and raises the internal temperature. We can also assume

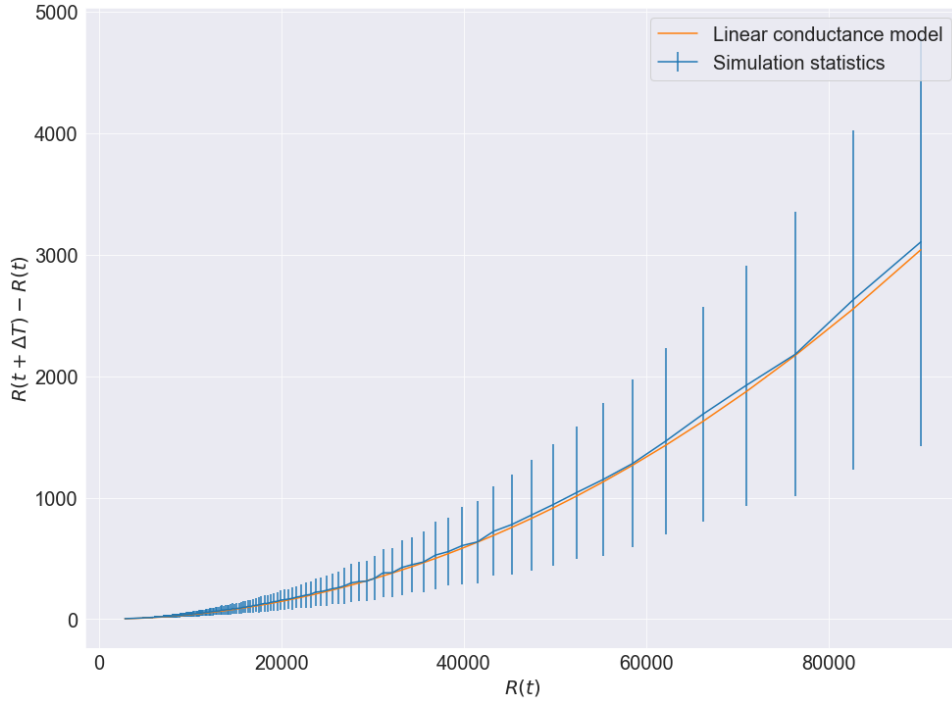


Figure 3.12: The mean change in the resistance following fitting to the linear conductance model, with errorbars indicating the standard deviation of series, for simulations with length $\Delta T = 10000$.

that Newton's law of cooling applies simultaneously, resulting in power dissipation and restoration of the internal temperature to the equilibrium (bath) temperature.

To introduce Joule heating as a form of volatility, one can define the rate of change of temperature in terms of a thermal time constant that is a function of a thermal resistance (R_{th}) and thermal capacitance (C_{th}) [Roldán et al. \(2021\)](#), dictating how easily heat energy in the form of temperature captures and is dissipated from the device, with the thermal time constant denoted $\tau_{th} = C_{th}R_{th}$. The differential equation for the internal temperature of the device is given as:

$$\frac{dT}{dt} = \left(T_{bath} + R_{th} \frac{V^2(t)}{R(n)} - T \right) / \tau_{th} \quad (3.22)$$

where $R(n)$ is the instantaneous resistance of the memristor (as a function of the state variable, n).

As an example of a form of additional volatility due to disruption of the structure, we assume that the voltage applied results in a bias-dependent equilibrium volatility, and the volatility approaches this value according to a differential equation, where the rate of change of the volatility is dependent on the difference between the instantaneous volatility and the equilibrium value. This observation is exploited to demonstrate the use of the volatility state variable in practice, in Section 3.5. The volatility and its evolution is defined by the following differential equation:

$$\frac{d\rho(t)}{dt} = (c_{\text{volatile}} \cdot V(t) - \rho(t)) / \tau_{\text{volatile}} \quad (3.23)$$

where τ_{volatile} is the volatility time constant, dictating how quickly the equilibrium value is attained (and therefore how slowly a non-zero volatility decays in the absence of an input signal); and c_{volatile} is the volatility factor, which - when multiplied by the applied bias - yields the equilibrium volatility.

3.4.5 Simulation

In addition to demonstrating the fit of the zero-bias mean values to those determined according to the linear conductance model, we demonstrate the capabilities of the model to simulate memristive behaviour through a series of simulated experiments for a titanium dioxide thin film memristive device.

3.4.5.1 Experimental Parameters

The parameters used for the experiments, chosen for $\Delta T = 10000$, are summarised in Table 3.3. The choices for these parameters are given in Sections 3.4.3.1 and 3.4.4.2.

Table 3.3: Table of parameters for the model, for titanium dioxide memristive devices.

Parameter	Description	Value
T_{bath}	Bath (equilibrium) temperature	300
$V_a = E_a/q$	Energy barrier height	0.40049
$V_{\text{off}} = E_{\text{off}}/q$	Difference in energy barrier heights at zero bias	0.05
N	Number of metastable switches	20,000
g_{parallel}	Parallel conductance for readout equation	10^{-10}
g_{step}	Conductance step (active region) for readout equation	10^{-10}
n_{thresh}	Threshold active switches for readout equation	10,000
$\tau_{\text{th}} = C_{\text{th}} \cdot R_{\text{th}}$	Thermal time constant for Newton cooling/-Joule heating	$3.84 \times 10^{-14} \cdot 4 \times 10^4$
c_{volatile}	The volatility state variable factor, dictating its impact on the switching rates	10
τ_{volatile}	The volatility state variable time constant	10

3.4.5.2 Switching Experiments

We use the model, with the derived parameters, to generate example device data for simulated input signals. Our first experiments assume there is no volatile behaviour. In the second set of experiments, we assume Joule heating - modulated by the current flowing through the device - and associated restoration of the equilibrium temperature according to Newton's law of cooling, as described in Section 3.4.4.3.

Table 3.4: Parameters for the input signals in the simulated switching experiments. In the case of the positive descending experiment, we observe that too large a signal amplitude drives the memristor immediately to a high resistance state, so we lower the initial amplitude to better observe the behaviour. In the case of the alternating experiment, we use a small duty cycle such that the active pulse duration is 1s.

Experiment	Max/Min Amplitude	Pulse Step	Pulse Duration (s)	Duty Cycle
Negative Ascending	-0.2	-0.01	1000	0.1
Negative Descending	-0.2	-0.01	1000	0.1
Positive Ascending	0.2	0.01	1000	0.1
Positive Descending	0.07	0.01	1000	0.1
Alternating	0.5	N/A	1000	0.001

We perform three sets of experiments: pulsing with positive voltage pulses (increasing/-decreasing in magnitude) to increase the device resistance, and with negative voltage pulses (increasing/decreasing in magnitude) to decrease the device resistance. We also examine the effect of alternating pulses. The results of these experiments are shown in Figure 3.13.

The input signal parameters are given in Table 3.4. We simulate series from starting resistances ranging between 10k Ω and 100k Ω . We choose the parameters for the input signals in order to highlight model behaviour. In all the experiments, we see that the model is biased towards movement to a higher resistance state, with negative pulses having a lesser impact on reducing the resistive state than positive pulses have on increasing the state. During the inactive period of each pulse, we see the device's resistance drift in the upwards direction as a result of the impact of V_{off} .

Note that for the purposes of these experiments, the effects of volatility in the form of Joule heating and other volatile effects such as structural disruption within the device are assumed to be negligible. In Section 3.5, we examine the potential for neuromorphic device behaviours to arise when the volatility parameters are such that volatile effects play a significant role.

3.5 NEUROMORPHIC APPLICATIONS

There are a number of ways that the model developed, including the notion of volatility, may shed light on neuromorphic approaches that aim to replicate the behaviour of biological neurons using memristors.

We may consider two types of memristor: those which are highly **unstable** and those that are **stable**. The stability of the memristor can be characterised by the rate of attainment of its equilibrium point, which is determined in large part by the magnitude of the energy barrier E_a . Less stable devices may be useful for the emulation of neuronal body dynamics, while those which are less volatile, which may be more useful for the emulation of synaptic dynamics. A system using a combination of both types of device may be necessary to implement the range of computational primitives available to neurons in the human brain.

STP can arise for the programming of an unstable memristor to a particular state while the volatility is high. The volatility may cause the state variable to partially revert to its equilibrium

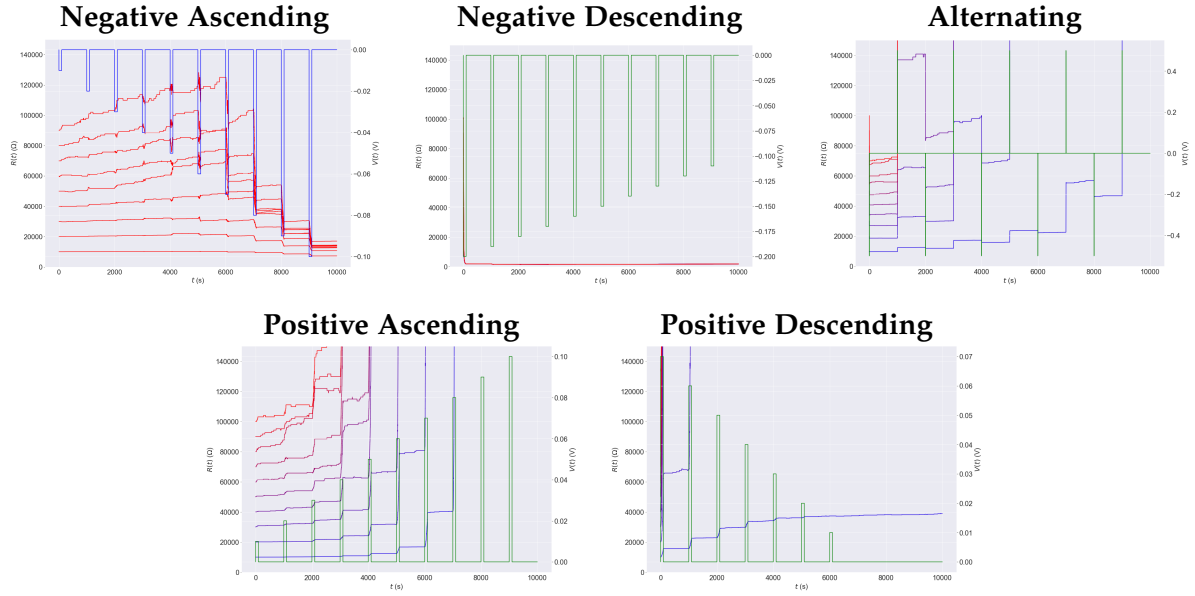


Figure 3.13: Simulated experiments for a variety of different stimulating voltage inputs. From the alternating pulses experiment, it is clear that the preferred direction of movement (the equilibrium resistance) being a high value, results in the final resistance being higher than the initial resistance, due to the impact of resistive drift, despite the same number and magnitude of positive and negative pulses being applied.

point, but eventually dissipates, resulting in stability of the resistive state (LTP). For these memristors, resistive drift is analogous to the leakage of membrane potential.

For more stable memristors emulating synaptic dynamics, the interplay between volatility and the application of a stimulating input (for example, current spikes) can result in self-limiting spiking responses. For more stable memristors, the resistive drift is analogous to the phenomenon of synaptic weight decay.

3.5.1 Synaptic Weight Decay through Resistive Drift

The phenomenon of resistive drift in stable memristive devices can be likened to the notion of synaptic weight decay. Under zero-bias conditions, we saw in Section 3.3.3, that the memristive state variable will drift towards an equilibrium value that is determined by the value of V_{off} , with a rate of attainment that is proportional to V_a .

For an array of memristors, once all of them reach the equilibrium state, due to the memory-less nature of the model (by virtue of its dependence on Poisson process switching) under zero bias, and assuming negligible volatile effects, almost all information about the previously programmed states will be lost. There is therefore a connection between the magnitude of (and the range of magnitudes possible for) the initial conductance and the lifespan of information that is represented by the value of the state variable. This relationship will be explored in detail in Chapter 5.

3.5.2 ReLU Nonlinearity for Memristive Neural Networks

The nature of the readout equation allows for the possibility of the implementation of nonlinear activation functions in the context of neural network acceleration on hardware. The output, as a function of the state variable - if nonlinear - can be used to apply said nonlinearity to any quantity used to represent information in the memristive neural network, particularly in dynamic neural networks, such as recurrent neural networks and spiking neural networks implemented in hardware.

For example, if the memristor is in a high resistive state, it may remain in this state until a sufficiently long applied voltage pulse drives it into the active region, as we observed may be possible in the case of a device that requires an electroforming step to drive it into the active region, where even a small change in the device's state changes the effective passive electrical characteristics (the instantaneous resistance in the simplest case). In our case, this thresholding behaviour, where the state variable does not impact the output conductance until it passes the value n_{thresh} , could potentially be used to implement the Rectified Linear Unit (ReLU) nonlinearity, which is a fundamental building block for deep neural network architectures.

3.5.3 Short and Long Term Memory

For stable memristors, the interplay between modifications to the absolute rates of switching caused by volatility, and biased switching effects from the application of voltage pulses of a given magnitude and time, can lead to memory effects, where states exhibit a dependence on time.

The inclusion of volatility in a memristive model adds a key characteristic that could prove crucial to the emulation of neuronal dynamics (which also includes leakage terms, sometimes described as part of their short-term memory) in addition to more persistent changes to the device resistance (long-term memory).

A sequence of voltage pulses can result in a very different final metastable resistance. For example, in the context of Joule heating, if the voltage pulses are short enough for adiabatic cooling (or relaxation) to occur, the volatility may not increase sufficiently to have an impact on the switching. If, however, voltage pulses are long enough, they may increase the volatility, resulting in metastable behaviour.

Metastability is a phenomenon that can be characterised by the volatility state variable. The volatility has an impact on both the **equilibrium point** and the rate of attainment of the equilibrium point (the **time constant of decay**). The impact of the volatility state variable on the equilibrium point for a titanium dioxide model was shown in Section 3.3.3.

This phenomenon can lead to changes in memristors that are dependent not only on the absolute energy supplied to a memristor, but also the pulse shapes and durations through which this energy is supplied, for example, resulting in dependence on the frequency of the pulses.

3.5.3.1 Frequency-Dependent Potentiation

In biological neurons, the generation of action potentials can be dependent on the frequency of the input, according to the leaky integrate and fire neuron model [Dayan and Abbott \(2001\)](#); [Gerstner et al. \(2014\)](#), where only an input spike frequency of a certain magnitude or greater may cause the generation of an action potential - a spike - within the postsynaptic neuron. As a result of the frequency-dependence of postsynaptic neuronal action potential generation and the mechanism of spike timing dependent plasticity (Hebb's rule) [Hebb \(1949\)](#), the potentiation of a neuronal synapse may also, consequently, be frequency-dependent. Frequency coding has been proposed as one of a number of coding schemes for the communication and processing of information in the human brain [Kandel \(2013\)](#).

In the context of memristive devices, the frequency of application of voltage pulses has been shown to have an effect on the number of pulses required to turn a device ON in biological memristors. This is attributed to the resistance drift back to the high resistance state during periods in which there is no input signal, meaning that higher frequency voltage pulses require fewer attempts on average to switch devices over a threshold [Fu et al. \(2020\)](#). The authors attribute the effect to the Fick's Law diffusion that occurs for the Ag⁺ ions in the ECM cell (the resistive drift). In another study of diffusive memristors - memristors which operate on the basis of the dispersal and regrouping of metal ion clusters under voltage bias [Wang et al. \(2017\)](#) - similar dependence of the final resistance of the devices was observed in relation to the frequency of programming pulses. It was suggested that this phenomenon could be used to emulate neuronal dynamics based on the transport of Ca²⁺ ions causing similar changes to the strength of synapses. Frequency-dependence of the number of pulses required to initiate a switching event has also been demonstrated in protein nanowire memristors [Iranmehr et al. \(2019\)](#). The number of pulses (of equal magnitude and duration) required to switch the devices from high resistive states to low resistive states was significantly reduced when the frequency of application was increased from 50Hz to 900Hz. A close to linear trend in the number of pulses required as a function of frequency was observed for frequencies higher than 200Hz, with increasing frequency reducing the number of total pulses required to initiate switching. However, a nonlinear trend was observed for sub-100Hz frequencies, where the number of pulses increased rapidly. This could evidence the presence of a transient volatility whose time constant of decay is on the order of $1/100\text{Hz} = 0.01\text{s}$.

The frequency-dependence of switching in memristors can be attributed to autonomous state evolution, analogous to the leaky term in the leaky integrate and fire model (the leak conductance in the Hodgkin-Huxley model of the neuron [Hodgkin and Huxley \(1952\)](#)). In the context of our event-based model, a device with an unstable switching characteristic (whose state restores to the equilibrium value quickly following the removal of applied bias) will exhibit such frequency dependent behaviour. This is dictated by the state the device is in, as well as the values V_a and V_{off} .

In Figure [3.14](#), we show how changes to the frequency of programming pulses can result in different short term and long term resistance memory effects in memristors due to the aforementioned interplay between the volatility and switching. For these experiments, we use

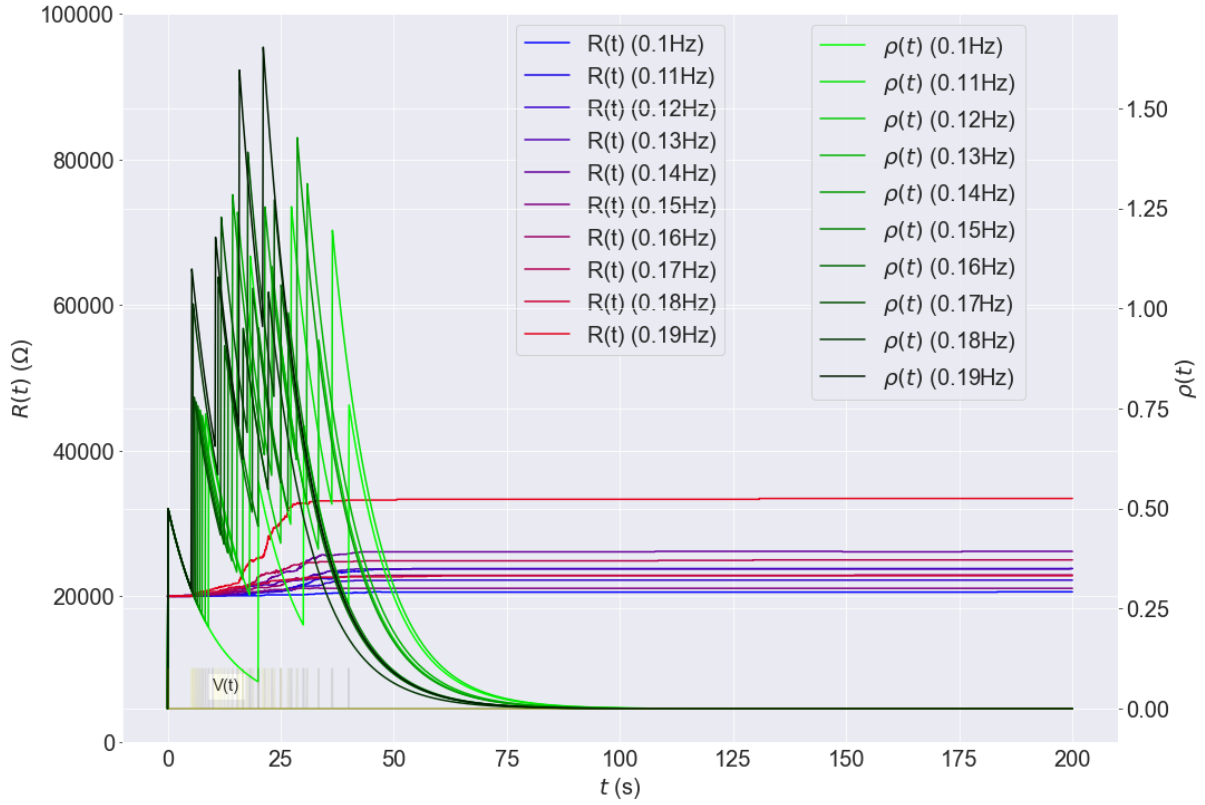


Figure 3.14: Frequency experiments for a starting state of $20k\Omega$: we see that despite the same total amount of energy being applied to the devices (the same magnitude voltage applied over the same total amount of time), we achieve a higher metastable resistive state when using a higher frequency input, corroborating frequency-dependent switching results observed in the literature. Note that this is a result of the larger magnitude attained by the volatility state variable, $\rho(t)$, for higher frequency inputs, which accelerates switching for a given bias.

a simulation length of $T = 100$, with intervals for updates of the volatility state variable ($\rho(t)$), of 0.1, with updates being generated according to the scheme described in Section 3.2.7.1.

In these experiments, we supply the same amount of energy to the devices (the same magnitude of voltage over the same amount of time in total), but demonstrate that if the impact of the volatility state variable is high enough, subsequent state transitions can differ greatly, dependent on the frequency of programming, due to the nonlinear acceleration in switching effected by an increase in the volatility state variable when the pulses are closely spaced, as compared to sparse pulses in time.

We apply square waves of period $T_{\text{pulse}} = 1/f$, where f is the frequency of the applied signal. Each square wave period contains a pulse of length 0.1s, and we supply 5 periods as an input signal in each experiment. As such, the same voltage magnitude is applied for the same total amount of time in each case.

We modify the parameter $c_{\text{volatile}} = 500$, keeping all other parameters as in Table 3.3.

3.5.4 Spiking Behaviour through Volatility

For less stable memristors, we explore a possible phenomenon enabled by the presence of volatility in a memristive system: the emulation of spiking action potentials. Such a phenomenon would be of use for synaptic emulation in neuromorphic circuits.

A “spike” can be defined in the context of a neuromorphic computing architecture as a short-lived increase (or decrease) in an electronic parameter that can be used to process information, such as the voltage, the current, or the conductance. A spike can be thought of as a self-limiting process: once a spike is initiated, it will effect its increase or decrease in the parameter, an associated effect of which is to cause the increase or decrease to cease after a short time. We can envision the creation of spiking behaviour in real devices, if we are able to engineer such a self-limiting process.

We here give a demonstration of how the principle of volatility can be used in the context of the proposed model to generate such a spike. Let us assume that the circuit consists of a memristor in series with a resistor, with an input signal applied across both the memristor and the resistor. This forms a voltage divider circuit, where the proportion of the applied voltage dropped across the memristor will be proportional to its effective instantaneous resistance. If the device is at a high resistance, the proportion of voltage dropped across the memristor will be large. Assuming a volatility that is proportional to the voltage dropped across the memristor (for example, one can think of Joule heating), the volatility state variable may increase as a result of the applied bias. An increase in the volatility state variable results in a drop in the equilibrium value of the memristor and an increase in the rate of attainment of this value (the net rate). As a result, the device resistance may drop to match the equilibrium point. However, as the resistance decreases, the voltage drops across the memristor, resulting in a decrease in the volatility, which in turn restores the memristor’s equilibrium point to its initial high resistance value, causing the device to return to the high resistance state. We can potentially use the current output of the circuit in order to read and pass on this spike to other parts of the neuromorphic architecture.

As we can see from Equation 3.12, the equilibrium state is a function of the volatility. An increase in the volatility tends to result in the movement of the equilibrium state towards the point $n_{eq} = N/2$, as the *up* and *down* switching rates become equal. Since the magnitude of both the *up* and *down* switching rates also goes up, so the rate of attainment of the equilibrium point goes up. This is visualised in Figure 3.15.

Biological neurons exhibit two responses to constant input stimuli: firing latencies or rates that are proportional to the magnitude of the input signal (time and frequency encoding, respectively), and a firing response that diminishes over time if the input signal remains constant (adaptation). We propose a method for producing such behaviour using the event-based model, focusing on the dynamics of the volatility state variable.

We can imagine a memristive device with a natural equilibrium state which corresponds to a low resistance. At the same time, we can imagine a volatility state variable which increases in response to the power dissipated by the device. At low resistances, the power dissipated in

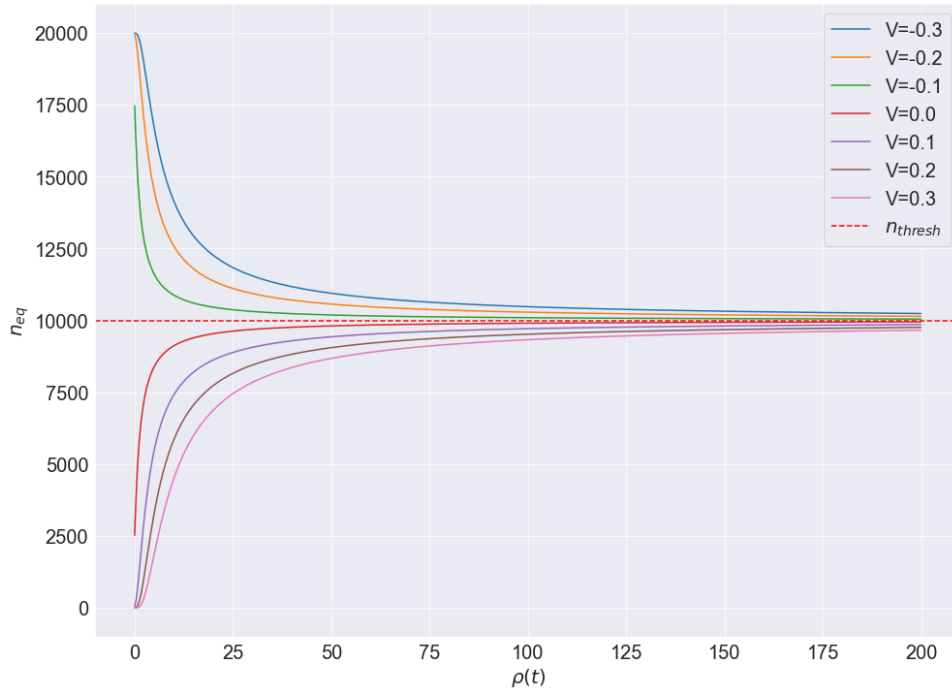


Figure 3.15: The impact of $\rho(t)$ on n_{eq} , for different bias values. Parameters for the calculation are drawn from Table 3.3. As $\rho(t)$ increases, the equilibrium point becomes closer to $N/2$.

Table 3.5: Parameter updates to produce spiking behaviour as shown in Figure 3.16, as compared to those in Table 3.3.

Parameter	Value
g_{step}	10^{-8}
$g_{parallel}$	10^{-6}
N	1000000
n_{thresh}	800000
V_a	1.0
V_{off}	-0.8

the memristor will be high, while at high resistances, the reverse will be true, if we assume the voltage drop across the device to be constant at all times.

If the device is at a low resistance, the dissipation of power will result in an increase in the volatility, which results in an increase in the equilibrium resistance (a drop in n_{eq}). As the device resistance rises in response, the power dissipated through the device drops. Due to the time constant of decay for volatility, the drop in the equilibrium state will not be instantaneous, and the device resistance will continue to rise. Finally, once the volatility decays sufficiently, the equilibrium resistance will restore to close to its original value, and the device will decrease in resistance, completing a “spike”. The number of such events that happen per unit time will be related to the magnitude of the applied signal. This is shown in Figure 3.16.

In order to produce such a spike train, several parameters were modified, as given in Table 3.5.

The behaviour is essentially brought about through a controlled oscillatory (self-inhibiting) - or transient - response.

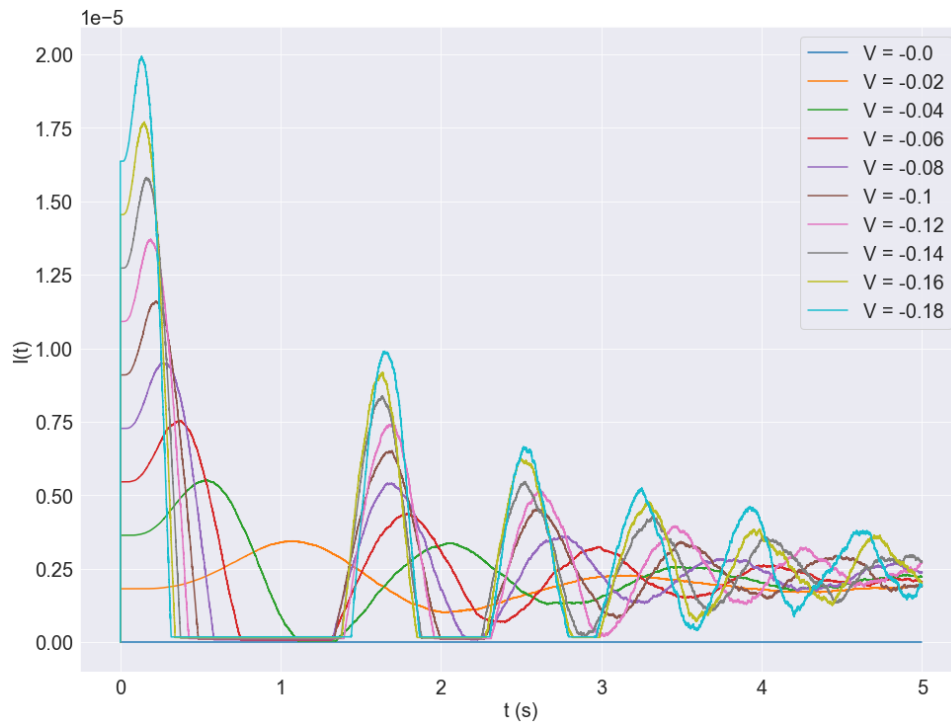


Figure 3.16: Spiking current behaviour produced using model parameters that produce oscillatory behaviour, made possible through the volatility state variable. Spiking behaviour can be produced through appropriate parameter selection. The intensity of the spikes, the number of spikes in the train, and the frequency of the resultant spike train are dependent on the magnitude of the applied input.

This example demonstrates how through consideration of the volatility state variable as a modelling tool, with its impact on the equilibrium state of the memristor and the rate of attainment of that state, we can predict memory-dependent, self-limiting, short-term behaviour such as spiking, on simulated memristors, offering motivation for its implementation on memristive hardware, through the tuning of device parameters.

Further work needs to be done on the characterisation of devices according to this framework, for example, through measurement of the impact of the volatility of a device on the equilibrium point and the rate of switching, in order to enable the practical implementation of such neuromorphic approaches.

3.6 CONCLUSIONS AND FUTURE WORK

In this chapter, we have introduced an event-based modelling strategy for memristive devices. We extended previous generic modelling approaches for memristors in order to address key shortcomings, namely: errors associated with discrete time approximation, and the lack of a clear connection to a generic physical model. To do this, we modified the modelling strategy to utilise Poisson processes, as opposed to binomial random variables, to model the rate of transition of metastable switches, as well as modifying the rate calculation to a Boltzmann

probability distribution. To accommodate nonlinear behaviour in the model, we used a readout equation to separate the state evolution dynamics from those of the parameters of interest, such as the instantaneous resistance or the voltage-current characteristic. Finally, we introduced a volatility state variable, to allow for the capture of memory-dependent switching behaviour, motivating this from a physical perspective for various memristive device types.

We also presented a resistive drift dataset, which we use to obtain parameters for an example of the model applied to a real device. We demonstrated the ability of the model to replicate the behaviour under zero bias.

Finally, we examined the modelling framework from the perspective of neuromorphic computing, studying possible requirements for parameters of future devices that may enable frequency-dependent switching, and the replication of action potentials using memristors.

Future work may include extending the modelling approaches described to additional real data from different memristive device types, under a variety of additional conditions, as well as exploration of the applications of the model for event-based neuromorphic systems and simulations, especially spiking neural networks implemented in hardware.

CONDITIONAL GENERATIVE ADVERSARIAL NETWORKS FOR MODELLING RESISTIVE DRIFT IN MEMRISTORS

The capacity of memristors for storing and processing information can be limited by noise. Resistive drift is an autonomous state evolution phenomenon that occurs in many memristive devices and causes the memristive state to change over time. We saw in Section 2.2.3 of Chapter 2 that resistive drift could take a variety of forms in different devices, and in Chapter 3, we attempted to find a unified model to describe them. We highlighted that resistive drift could be likened to the phenomenon of synaptic weight decay in memristors, or alternatively - for unstable memristors - to neuronal membrane leakage (see Section 2.1 of Chapter 2). It is clear that resistive drift has the potential to be exploited as a computational primitive for neuromorphic memristive systems. Motivated by the potential applications of resistive drift, with its unique property of being a memristive nonideality whose uncertainty compounds over time instead of space, we look to model the drift with a view to investigating the limits of information storage on memristors in the presence of this type of noise.

In this chapter, we extend existing communication-theoretic approaches to the problem of information storage on memristors in the presence of nonidealities, as introduced in Section 2.4.2 of Chapter 2. To this end, we employ the approach of generative channel modelling in order to learn a differentiable channel model of the resistive drift, using a form of Generative Adversarial Networks (GANs), called conditional Generative Adversarial Networks (cGANs), to model the conditional distribution Ding et al. (2022); Goodfellow et al. (2014); Mirza and Osindero (2014); Ye et al. (2018, 2020). In the context of resistive drift, the conditional probability distribution has two continuous conditions: the initial resistance and the delay. Although we have already developed a general event-based model in the previous chapter, the approach of using generative modelling to simulate the resistive drift has the advantage of being purely data driven, and allows us to model the drift over time in a computationally efficient manner, creating a differentiable model that is suitable for use in deep learning frameworks, reducing computational load and problems associated with vanishing gradients.

We make use of the model developed in Chapter 3 to generate a sample stochastic resistive drift dataset, with a known equilibrium resistance, and employ the use of a cGAN to model the conditional distribution, by training on this dataset. We condition on the initial resistance and, in an effort to reduce the computational complexity of simulation, introduce the use of techniques - including a novel “delay discriminator” approach - to allow for conditioning on a wide range of delays, in order to reduce the number of recurrent evaluations necessary to compute the conditional channel distribution for a given delay.

Notation Throughout this chapter, we use $\text{BCE}(y, \hat{y}) \triangleq (y - 1) \ln(1 - \hat{y}) - y \ln(\hat{y})$ to represent the binary cross-entropy function. In general, we use lowercase r to denote resistance values, d to denote delay values, and $\bar{y}$ to denote a normalised version of a given variable y . $N_x(\cdot)$ is used to denote a given normalisation transform.

4.1 INTRODUCTION

There are many choices of generative network that could be used to simulate data distributions, as mentioned in Section 2.5. Time series are a special kind of data modality, with temporal correlations and dependencies, which require specialised modelling techniques.

Time series generative models come in a variety of forms. They can be either recurrent, or non-recurrent. Recurrent architectures make use of stateful neural networks such as Long Short-Term Memory (LSTM) cells Hochreiter and Schmidhuber (1997) or Gated Recurrent Units (GRUs) Cho et al. (2014), which use a state variable to capture dependencies between elements of the series over time. Alternatively, finite-memory time series generation can be achieved through the use of modern autoregressive model architectures van den Oord et al. (2016), which generate values by feeding a neural network a fixed horizon of past inputs, extending earlier autoregressive modelling approaches which modelled outputs as linear combinations of past inputs.

Despite the effectiveness of autoregressive models, they are fundamentally deterministic, in the sense that they will always produce the same output value given the same values for past inputs. In addition to this, problems in training recurrent time series models may arise due to the disconnect between so-called “closed-loop” training and “open-loop” evaluation Yoon and Jarrett (2019), where the contrast between evaluation of the model in a recurrent manner during inference and its conditioning on ground truth data during training may cause problems with distributional mismatch. Several methods exist for addressing this problem, notably the use of scheduled sampling (a form of curriculum learning), involving a gradual transition between open-loop training and closed-loop training Bengio et al. (2015). However, it has been suggested that techniques such as scheduled sampling are insufficient for addressing this problem when used alone Yoon and Jarrett (2019), with the technique also being criticised for memorisation of samples Huszár (2015).

Challenges associated with generating time series using recurrent neural network architectures include the computational complexity of the modelling, with many subsequent sampling steps leading a large number of layer forward passes, if the recurrent architecture is unrolled. This increases the computational complexity of the generation of a data sample and also means that the network is prone to suffer from issues such as the vanishing gradient problem during training, or if used as a fixed differentiable model in an end-to-end training setup. In our context, future values are dependent only on the current state, thus the inclusion of a state variable should not be necessary, reducing the computational complexity.

As an alternative to supervised recurrent or autoregressive models, GANs have been shown to be capable of effectively modelling complex input distributions probabilistically, through the use of adversarial learning. Time Series GAN is a GAN trained to generate time series data, which has shown success in the accurate modelling of information over time Yoon and Jarrett (2019). The authors suggest that adversarial feedback may not be sufficient to capture the temporal dynamics present in time series and proposed the introduction of an additional supervised loss to enable the model to exploit the temporal correlations and structure in time series as an inductive bias for more realistic time series data generation.

As part of our work, we examine the use of continuous delay-conditioning for improving the training of **cGANs** used to generate time series data, investigating a novel technique for incorporating the inductive bias of temporal correlation between series values into the network’s training procedure, through the use of an approach that augments the adversarial loss, rather than introducing a supervised term. Namely, we make use of an additional critic in the adversarial training that we call the “delay discriminator”, as a technique for introducing conditioning on the continuous delay condition, and enforcing consistency in generation across timescales.

Many traditional time series generative models are limited in the timescales and horizons at which they can generate or predict time series data, meaning they can only generate values at integer multiples of the sampling rate, and for fixed future time horizons. Techniques such as fractional delay filtering can be used to predict or generate data for delays that are not integer multiples of the sampling rate through interpolation. Other techniques can generate samples with no fixed upper limit on the delay [De Vries and Principe \(1992\)](#), however they may exhibit poor performance for prediction of samples far into the future, not capturing the complete distribution of possible outputs for larger delays.

In conditioning on the continuous delay, we avoid the need for many recurrent evaluations of the model, making the model computationally efficient, and useful as a differentiable model for use in the training of deep neural networks. This can also simplify the training for other networks which make use of the trained **cGAN** as a fixed model, as we will explore in Chapter 5. Delay conditioning allows the model to generate outputs at arbitrary continuous delay values, rather than being constrained to generate values at multiples of the smallest timestep, meaning that model outputs can be obtained for delays at a range of orders of magnitude, including both delays that are non-integer multiples of the smallest timestep in the training data and delays that are arbitrarily large. The proposed approach allows for modelling a rich distribution conditioned on previous inputs and the given delay, generating not only a single prediction, but varied samples that jointly capture the variance and higher order moments of the conditional distribution of future series values.

In Section 4.2, we choose parameters for and generate a simulated resistive drift dataset, making use of the memristor model developed in Chapter 3. In Section 4.3, we introduce the network training procedure and methods, including the novel approach of delay discrimination, which is used to enable our network to learn to generate outputs conditioned on a wide range of delays. We present the results of the model fitting in Section 4.4.

4.2 RESISTIVE DRIFT DATASET

In order to generate a drift dataset suitable for use in training our generative resistive drift model, we make use of the event-based memristor model developed in Chapter 3. We generate an exemplar memristor drift dataset consisting of 5000 datapoints [El-Geresy et al. \(2024b\)](#). The event-based model parameters are chosen to simulate a particularly stochastic memristive device, to present a challenging learning setting.

Table 4.1: The parameters used for generation of the simulated resistance drift dataset, with an equilibrium resistance of 500k Ω .

Parameter	Value
N	1×10^6
V_a	0.256
n_{thresh}	0
R_{high}	1×10^6
R_{low}	1×10^2
V_{off}	0.2533
T	300
k_B	1.38064×10^{-23}
q	1.602176×10^{-19}

4.2.1 Parameter Selection and Dataset Creation

We choose the parameter V_a , controlling the rate of switching, to yield a drift distribution with sufficient change and stochasticity over the interval of interest of 1000s. We choose the parameters N, n_{thresh} , g_{step} , and g_{parallel} to allow for a sufficiently fine description of the states over the resistance interval [0.1k Ω , 100k Ω]: our chosen active switching region.

We choose our equilibrium point to be approximately 500k Ω , corresponding to $n = 100$ according to the chosen parameters and the chosen readout equation, mimicking that developed for the titanium dioxide memristor in Section 3.2.3 of Chapter 3, shown in Equation 4.1.

$$R(n) = \frac{1}{g_{\text{diff}} \max(n, n_{\text{thresh}}) + \frac{1}{R_{\text{high}}}} \quad (4.1)$$

We set V_{off} accordingly to achieve this, using Equation 4.2, as derived in Section 3.3.3 of Chapter 3. Using a value of $n_{\text{eq}} = 100$ along with the chosen value of $V_a = 0.256$, in order to adjust the stability/speed of convergence, gives a required value of $V_{\text{off}} = 0.2533$ to achieve the necessary equilibrium state.

$$V_{\text{off}} = \ln \left(\frac{N - n_{\text{eq}}}{n_{\text{eq}}} \right) \cdot \frac{k_B T}{q} \quad (4.2)$$

For the given V_{off} , we invert the equation to calculate the error in our equilibrium point: we can see that this yields gives a true equilibrium value of $n_{\text{eq}} = 99.993$, corresponding to a resistance of $R_{\text{eq}} = 500.042\text{k}\Omega$, which is sufficiently close to the desired value of 500k Ω .

A summary of the physical parameters and constants chosen for generation of the data are given in Table 4.1.

We generate 5000 series with initial points spaced uniformly in resistance in the range [100 Ω , 300k Ω]. For each generated series, we resample the non-uniform sampled data according to sampling Algorithm 3 from Chapter 3, with $T_{\text{sample}} = 1$ and $T_{\text{tot}} = 1000$, resulting in 1001

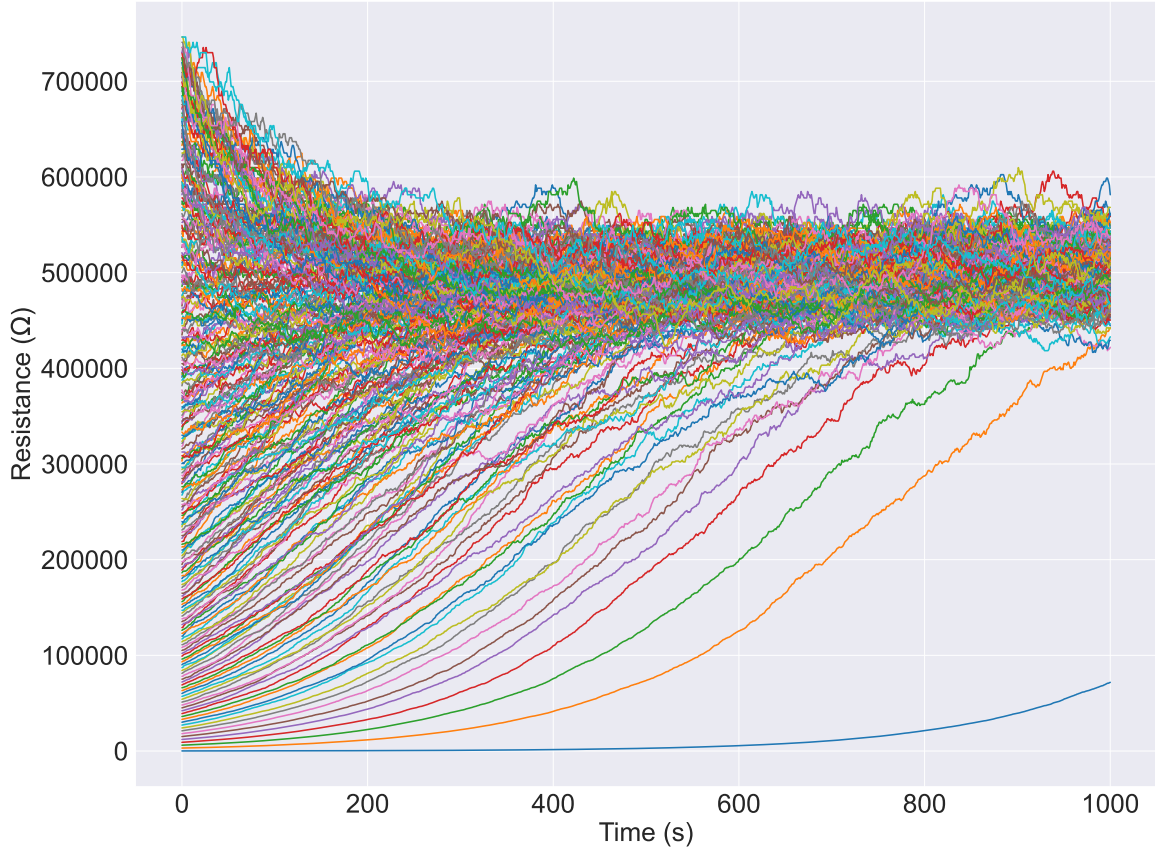


Figure 4.1: An illustrative subset of the series in the dataset used for training the [cGAN](#), generated using the event-based model of Chapter 3, used to illustrate the stochasticity and the convergence to the equilibrium point of 500kΩ. Note the increase in stochastic nature of the drift as a function of the magnitude of the resistance.

data points in each series, spaced one second apart (inclusive of the boundaries). We generate the initial resistance values (and corresponding values of n), according to a set of resistances chosen uniformly from the range $[100\Omega, 300\text{k}\Omega]$ (inclusive).

4.2.2 Dataset Visualisation and Evaluation

All series in the resistive drift dataset generated are shown in Figure 4.1.

The distribution of final resistances for each series in the dataset is shown in Figure 4.2, showing that over the course of 1000 seconds, almost all series converge to the region of the equilibrium resistance of 500kΩ. This demonstrates that the information capacity of the device is a function of the delay expected for storage, with the information capacity decreasing over time. As can be seen from the figure, an imbalance in the energy levels of the “high” and the “low” states contributes to the phenomenon of resistance drift towards an equilibrium point (even in the absence of a bias voltage) that is temperature and bias -dependent. In our case, the energy imbalance favours drift in the direction of the high resistance. This stochastic deviation

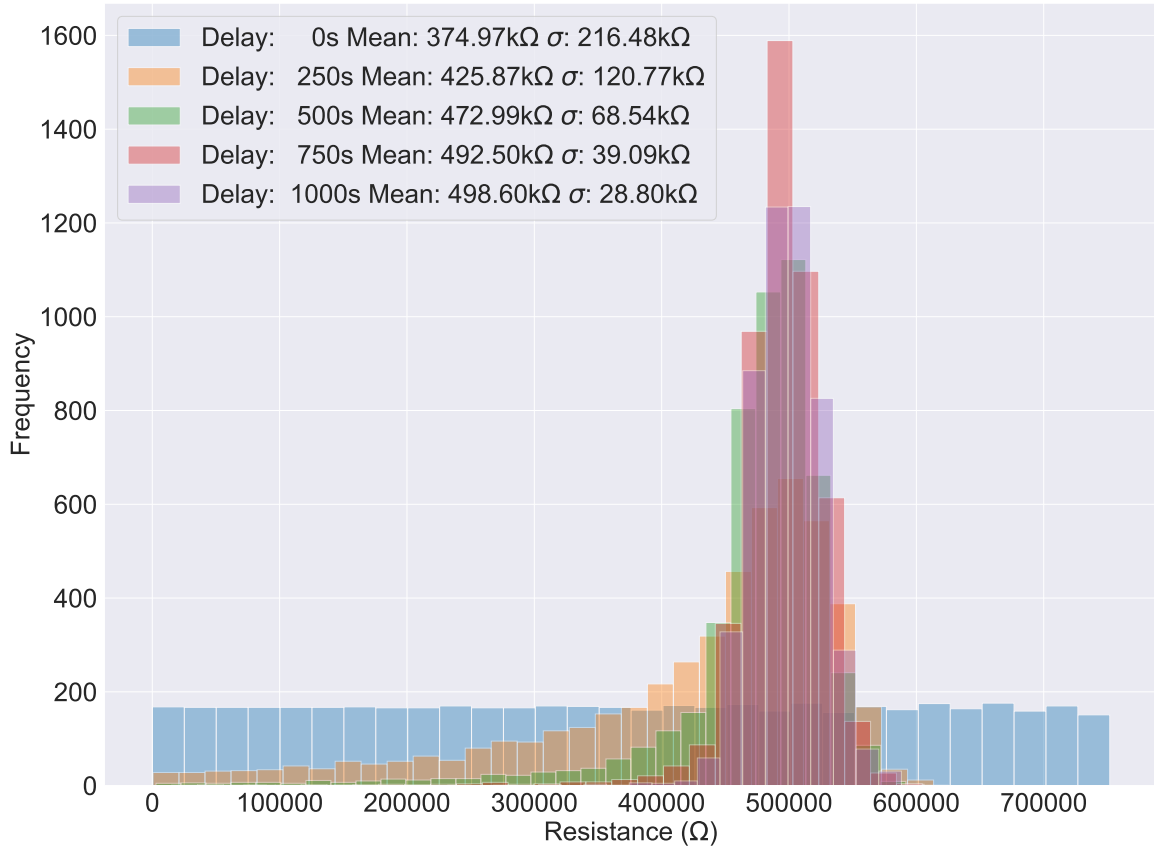


Figure 4.2: Histogram of final resistance values in the drift dataset at different delays. We see that as the delay increases to 1000s, the distribution becomes centred around the equilibrium resistance value - i.e. any information present in the encoding of the initial value is lost. The legend shows the mean value for each delay conditioned distribution, μ , and the standard deviation, σ .

of the resistance from its initial point is the noisy distribution that we wish to model using the Conditional GAN.

4.3 CONDITIONAL GAN TRAINING

There were three main challenges associated with training the network on the resistive drift dataset.

1. **Data scaling:** the wide dynamic range of the series values meant that simple data normalisation or batch normalisation were not effective.
2. **Delay consistency:** accurately conditioning on the delay requires the incorporation of the inductive bias that the network's output be consistent across different delay conditions.
3. **Open-loop, closed-loop disconnect:** training that occurs in closed-loop mode, where series values are not fed back, can result in poor performance in open-loop evaluation due to compounding errors not highlighted during the training process.

These three problems were addressed using appropriate data transformations, as well as the use of a novel “delay discriminator” approach to training models to output data that is consistent across timescales, also enabling the model to generalise beyond the domain of the delays in the training set.

4.3.1 Generative Adversarial Network Training

We use a canonical Binary Cross-Entropy (BCE) loss for the discriminator for our GAN training. We denote the generator network as G_{θ_g} , parameterised by θ_g ; and we denote the main discriminator network as $D_{d;\theta_d}$, parameterised by θ_d Goodfellow et al. (2014). Let us denote the distribution induced¹ by sampling from G_{θ_g} as $p_f(f)$, and the ground truth data distribution as $p_x(x)$.

Our discriminator objective is the maximisation of the likelihood of classifying samples as 1, or “real”, when evaluated on real data samples, and the simultaneous maximisation of the likelihood of classifying samples as 0, or “fake”, when evaluated on the distribution induced by the generator. This is achieved by the discriminator’s minimisation objective being:

$$L_{\theta_d} = -\mathbf{E}_{x \sim p_x(x)} [\log (D_{d;\theta_d}(x))] - \mathbf{E}_{f \sim p_f(f)} [\log (1 - D_{d;\theta_d}(f))] \quad (4.3)$$

And our generator objective is to maximisation the likelihood of the discriminator classifying samples generated from the induced distribution as 1, or “real”. This is achieved by the generator’s minimisation objective being:

$$L_{\theta_d} = -\mathbf{E}_{x \sim p_x(x)} [\log (D_{d;\theta_d}(x))] \quad (4.4)$$

Wasserstein GAN proposes an alternative optimisation objective, requiring the use of the associated techniques of discriminator weight or gradient clipping to enforce the Lipschitz constraints on the discriminator Arjovsky et al. (2017). However, we found that for our problem setting, the original GAN loss function as described in Equations 4.3 and 4.4 above, yielded better convergence.

4.3.2 Data Transformation and Normalisation

For any data-driven optimisation problem, it is important to appropriately transform the input data, in order to incorporate known inductive biases that facilitate learning. It is also well-known that data normalisation aids the learning process by ensuring that there is no bias in the contribution of features to the gradients of the network during optimisation.

The nature of our input features - the resistances and the delays - presents a challenge to the learning, due to the large range of values which they took on. We must choose an

¹ A generative model implicitly defines a distribution through a combination of its parameters and its source of randomness, as explained in Section 2.5 of Chapter 2.

appropriate data transformation that reduces the large range, while also allowing for precision in generation at a variety of orders of magnitude.

In the case of the normalisation of the resistance values, we apply a logarithmic transform followed by subtraction of the mean and scaling by the standard deviation (using statistics empirically averaged over the drift dataset) in order to normalise the magnitudes. We determine the parameter μ_R as the sample mean of the logarithmically transformed resistances, taken over the dataset, and similarly find σ_R as the square root of the sample variance of the resistances present in all resistance series, S_i - indexed by i , in the dataset, D :

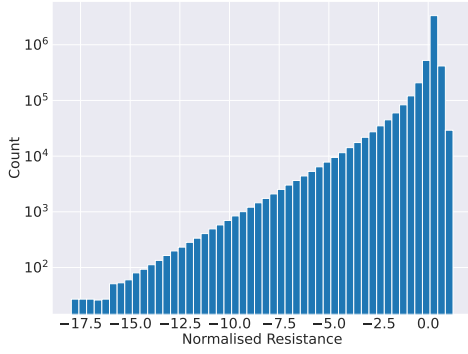
$$\begin{aligned}\mu_R &= \frac{1}{|D|} \sum_{S_i \in D} \frac{1}{|S_i|} \sum_{r_j^i \in S_i} \ln(r_j^i) \\ \sigma_R &= \sqrt{\frac{1}{|D|} \sum_{S_i \in D} \frac{1}{|S_i|} \sum_{r_j^i \in S_i} (\ln(r_j^i) - \mu_D)^2}\end{aligned}\quad (4.5)$$

where $r_j^i \in S_i$ are the ordered (resistance) elements of the series S_i , indexed by j . Thus, the resistance normalisation transformation, $N_{\text{res}}(r)$, is given as:

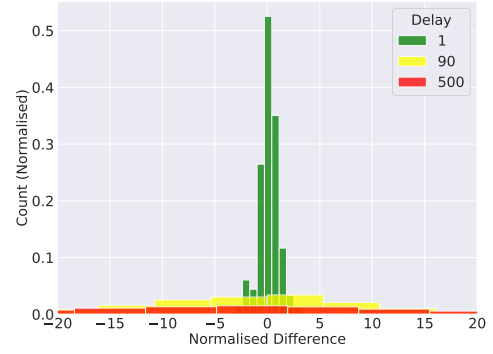
$$N_{\text{res}}(r) = \frac{\ln(r) - \mu_D}{\sigma_D} \quad (4.6)$$

We found that it was more efficient to train the generator network to predict the difference in the normalised resistance value, rather than the absolute value. As such, we apply a residual connection [He et al. \(2016\)](#) between the input and the output. Prediction of the difference encodes the inductive bias that the output should be equal to the input by default, and is a common feature of popular neural network architectures, such as U-Net [Ronneberger et al. \(2015\)](#), or multi-head attention modules in transformers [Vaswani et al. \(2017\)](#).

To normalise the differences of the normalised resistances, we apply a conditional mapping - $N_{\text{diff}}(\cdot, \cdot)$ - that is dependent on both the normalised input resistance, $\bar{r}_{\text{init}} = N_{\text{res}}(r_{\text{init}})$, and the normalised output resistance, $\bar{r}_{\text{final}} = N_{\text{res}}(r_{\text{final}})$, resulting in the normalised difference, denoted by $\bar{r}_{\text{diff}}$. Note that the differences normalisation transform is applied atop the data normalisation transformation. Let $\bar{D}$ denote the normalised dataset, created by applying $N_{\text{res}}(\cdot)$ pointwise to all $r_j^i \in S_i \forall S_i \in D$. The forward (Equation 4.8) and inverse (Equation 4.7) difference transforms are defined as follows, with the standard deviation used for normalisation calculated from the dataset statistics for a delay of 1:



(a) Histogram of the normalised resistances, obtained by applying the transform $N_{\text{Res}}(\cdot)$ pointwise to the resistances in the series of the dataset D.



(b) Normalised histograms for the differences, for a range of delay conditions, derived empirically from the dataset D, obtained by applying the differences transform $N_{\text{diff}}(\cdot, \cdot)$ to the pairs of values separated by delay ΔT .

Figure 4.3: Histograms of the values in the dataset, D following the initial pointwise application of the resistance normalisation transform (Figure 4.3a) and subsequent application of the differences transform for pairs of values from the dataset separated by a given delay (Figure 4.3b).

$$\mu_{\bar{D}} = \frac{1}{|\bar{D}|} \sum_{S_i \in \bar{D}} \frac{1}{|S_i| - 1} \sum_{\{r_j^i \in S_i | j < |S_i|\}} (r_{j+1}^i - r_j^i)$$

$$\sigma_{\bar{D}} = \sqrt{\frac{1}{|\bar{D}|} \sum_{S_i \in \bar{D}} \frac{1}{|S_i| - 1} \sum_{r_j^i \in S_i} (r_j^i - \mu_{\bar{D}})^2}$$

$$N_{\text{diff}}^{-1}(\bar{r}_{\text{diff}}, \bar{r}_{\text{init}}) = \bar{r}_{\text{init}} + \bar{r}_{\text{diff}} \cdot \sigma_{\bar{D}} \quad (4.7)$$

$$N_{\text{diff}}(\bar{r}_{\text{final}}, \bar{r}_{\text{init}}) = \frac{(\bar{r}_{\text{final}} - \bar{r}_{\text{init}})}{\sigma_{\bar{D}}} = \rho \quad (4.8)$$

Figure 4.3 shows the distribution of values following the application of the resistance normalisation and differences normalisation transforms to the resistances in the dataset. In both cases, the normalisation transforms map the data to a more reasonable input range, while also retaining precision in their descriptive capabilities at different orders of magnitude. Note that in the case of the differences transform, larger delays result in a range observed in the transformed histogram. It was also possible to ensure that the histograms were limited in their range by incorporating delay scaling in the differences normalisation transform, however, we found that retaining the different ranges of the transformed resistances for different delay conditions actually improved the training performance.

Similarly, despite the large range of delays used for training, denoted by $[d_{\min}, d_{\max}]$ with $d_{\min} = 1$ and $d_{\max} = 500$, it was found that scaling of the delays using a logarithmic transform did not benefit training in the case of the cGAN. This is likely due to the magnitude of the normalised differences (the feature that the generator is trained to predict) being somewhat

proportional to the delay condition. Thus, we apply no transformation to the delays and pass the unscaled delays directly to the network.

4.3.3 Feeding Data to the Network

We experimented with the choice of which transformed versions of the input to pass to each of the networks - the generator and the discriminator(s). We found that presenting a vector consisting of both the normalised resistances, $\bar{r}_{\text{init}}$, and the normalised differences, $\bar{r}_{\text{diff}}$, to the discriminator improved the performance. The normalised resistances can be seen as providing information about the absolute value of the input (the order of magnitude), while the normalised differences provide more fine-tuned information about the relative values. The discriminator is fed a series of s normalised resistance values simultaneously, along with a series of $s - 1$ normalised difference values, that are obtained by computing the differences transform of the pairs of resistance values. The first element of the resistance series fed to the discriminator is always a real datapoint, and the other $s - 1$ values are either the remainder of the series generated from the training set, or $s - 1$ generated values. The output of the discriminator is a real number in the range $[0, 1]$ representing the belief that the given resistances and delay were drawn from the dataset (1) or that they were generated (0). The dimensionality and the nature of the inputs to the discriminator is visualised in Figure 4.5.

Although presenting the normalised differences to the discriminator resulted in improvements to the training, we did not pass the normalised resistances as inputs to the generator, in order to alleviate the need for more than one initial data point to condition the generator, allowing it to output a sample from a conditional distribution (and subsequent recurrently evaluated samples) given a single resistance conditioning value r_{init} . In the generation of resistance series, we use no state variable, conditioning the generator only on the previous resistance value, implicitly incorporating the inductive bias that the resistance series are memoryless. The generator input is thus a single scalar variable representing a normalised resistance, along with a single scalar delay condition. We apply an inverse differences transform at the output of the generator, along with a residual connection to add the resultant difference to the normalised input resistance to produce a normalised generated final resistance. The dimensionality and nature of the inputs to the generator can be seen in Figure 4.4.

4.3.4 Multiple Sample Discrimination

One notable problem that plagues the training of GANs is the problem of mode collapse [Arjovsky and Bottou \(2016\)](#). In our case, the problem presents itself as a tendency for the network to accurately learn the mean value of the resistance changes, but to learn a variance that is smaller than the that of the true conditional distribution.

In order to improve the discriminator's effectiveness at identifying this phenomenon in an attempt to alleviate mode collapse, we pass n real or generated sequences jointly to the discriminator as input. Thus, our effective batch size is reduced to b/n for the discriminator.

This technique has been previously demonstrated to improve training by addressing the problem of mode collapse [Lin et al. \(2020\)](#); [Salimans et al. \(2016\)](#).

The disadvantage of this technique, in addition to the reduction in the size of the batch, is that it increases the effective dimensionality of the discriminator input. We must therefore trade-off these disadvantages for the reduced level of mode-collapse observed with this increased network computational complexity. In practice, we find a value of $n = 2$ produces good results and that is the setting used for the results presented in this chapter. In order to see the results of an ablation experiment using a value of $n = 1$ (i.e. no multiple sample discrimination), we refer the reader to Appendix [A](#).

This technique is applied to both the main discriminator network, and the auxiliary delay discriminator network, that will be described in the next Section [4.3.5](#). The use of multiple simultaneous samples for input to the discriminator is shown in the a diagram depicting the discriminator architecture in Figure [4.5](#).

4.3.5 Delay Discriminator

As discussed in Section [4.1](#), previous research has highlighted problems associated with training GANs adversarially to generate time series data.

We overcame the need for using scheduled sampling, representation learning, supervised loss functions, or recurrent networks, and were able to train a non-recurrent generative network by conditioning it on a continuous delay condition, through enforcing consistency in the model’s generation at various timescales.

This was a desirable target, since it would help alleviate the vanishing gradient problems associated with recurrent neural network architectures, which - in turn - arise from their deep structure when unrolled. Indeed, it has been shown that recurrent neural networks do not have an infinite memory in practice, and recursion can often be replaced by feed-forward networks [Dauphin et al. \(2017\)](#). Since the purpose of our model is to be used as a proxy for a physical imperfection in the data storage problem, for example in an end-to-end differentiable learning setup, it was desirable to have a model that was as simple and shallow as possible, to enable the training of an end-to-end system (we will explore this further in Chapter [5](#)).

Our approach involves two aspects. The first of these is training in a mixed-mode setup, with the condition being passed as an initial value and the subsequent values fed recurrently back to the network to generate subsequent samples. Once a sequence of $s - 1$ additional values have been recurrently generated by the generator network, the entire generated sequence is fed simultaneously to the discriminator. This ensures that both open-loop and closed-loop errors are used simultaneously to update the network parameters.

However, this alone does not allow for the network to effectively learn to condition on a continuous delay condition. As such, we introduce a novel approach known as “delay discrimination” using a “delay discriminator”. The delay discriminator is an auxiliary discriminator network trained alongside the main discriminator. We use this to simultaneously address the closed-loop training open-loop evaluation disconnect, and also to improve the performance for larger delay conditions, outside of those available in the original training data.

We base our method on the inductive bias that a network conditioned on the delay should generate the same output distribution, regardless of the magnitude of the delay, if evaluated recurrently, so long as the total delay is the same. The delay discrimination algorithm is shown in Algorithm 4 for a single batch size of b , where $D_{dd;\theta_{dd}} : \mathbf{R}^2 \rightarrow [0, 1]$ represents the delay discriminator mapping, parameterised by θ_{dd} .

The principle is to train an auxiliary discriminator network to differentiate between the final values of two model series trained for different timescales. We first randomly generate a delay discriminator delay in the range $d_{dd} \in [d_{dd\min}, d_{dd\max}]$, where $d_{dd\min} > 0$ and $d_{dd\max} \geq d_{\max}$, as well as a random integer factor in the range $q \in \{2, \dots, q_{\max}\}$.

We first condition the generator on the delay d_{dd} and, given a randomly selected resistance r_{dd} as the initial condition, generate a single sample. Since this sample is generated with no reference to previously generated samples, it does not suffer from closed loop errors. Following this, we generate q samples using the generator, with a delay condition of d_{dd}/q . The result of this training process is to cause the generator to change its generation of the series of delay d to be more akin to the series with delay d/q and vice versa. The procedure for training the delay discriminator is shown in Algorithm 4.

Algorithm 4 Algorithm for updating the network parameters of the generator and delay discriminator for a single batch of size b . α represents the learning rate of the optimisation strategy. Note that we show the algorithm here for a value of $m = 1$ according to multiple sample discrimination, in order to simplify the notation and improve clarity. In practice, we use a value larger than 1.

```

1:  $q \sim \mathcal{U}\{2, q_{\max}\}$ 
2:  $\mathcal{P}_{dd,1} \leftarrow \{\}$ 
3:  $\mathcal{P}_{dd,2} \leftarrow \{\}$ 
4: for  $i \in b$  do
5:    $d_{dd} \in \mathcal{U}\{d_{\min,dd}, d_{\max,dd}\}$ 
6:    $r_{\text{init}} \leftarrow \mathcal{U}\{R_{\min}, R_{\max}\}$ 
7:    $r_{\text{final},1} = G_{\theta_g}(r_{\text{init}}, d_{dd})$ 
8:    $r_{\text{final},2} \leftarrow r_{\text{init}}$ 
9:   for  $j \in q$  do
10:     $r_{\text{final},2} \leftarrow G_{\theta_g}(r_{\text{init}}, d_{dd}/q)$ 
11:   end for
12:    $\mathcal{P}_{dd,1} \leftarrow \mathcal{P}_{dd,1} \cup D_{dd;\theta_{dd}}(r_{\text{init}}, r_{\text{final},1})$ 
13:    $\mathcal{P}_{dd,2} \leftarrow \mathcal{P}_{dd,2} \cup D_{dd;\theta_{dd}}(r_{\text{init}}, r_{\text{final},2})$ 
14: end for
15:  $\theta_{dd} \leftarrow \theta_{dd} + \alpha \nabla_{\theta_{dd}}(\text{BCE}(\mathcal{P}_{dd,1}, \{0, \dots, 0\})) + \alpha \nabla_{\theta_{dd}}(\mathcal{P}_{dd,2}, \{1, \dots, 1\})$ 
16:  $\theta_g \leftarrow \theta_g + \alpha \nabla_{\theta_g}(\text{BCE}(\mathcal{P}_{dd,1}, \{1, \dots, 1\}))$ 

```

We discard all but the conditioning resistance and the last sample for this second generated series. The delay discriminator, $D_{dd;\theta_{dd}}$, is trained to distinguish between final sample of the recurrent, open-loop, evaluation for the smaller delay, and the sample generated using a single (non-recurrent) generation step for the larger delay. As such, this benefits the training in several ways:

1. Information from the closed-loop evaluation for larger delays passes through to condition the open-loop evaluation for smaller delays, allowing inconsistencies to be highlighted and connecting the two forms of evaluation.
2. The inclusion of the delay consistency inductive bias, that reduces the possible model outputs, reducing training complexity and improving convergence.
3. The inclusion of delays larger than those possible to generate in the dataset, as well as fractional delays, allowing for a much wider range of delays to be used for conditioning the generator in the training process. Following training, the generator can model the conditional distribution after much larger delays, without many recurrent evaluation steps.

We will highlight the effectiveness of the delay discriminator method in Section 4.4.2, where we look at the mean and variance of the values after an equivalent total delay of 1000 for series generated using delay conditions in the set $[5, 10, 100, 250, 1000]$. We will demonstrate that using the delay discriminator vastly improves consistency of generation - that is to say that the distribution of the model outputs will always be more similar when conditioning on the same total delay, independent of whether the output is directly conditioned on the total required delay, or instead generated in a recurrent fashion using multiple steps each at fractions of the total delay. This is especially significant for larger delay conditions.

4.3.6 Architecture

We use a modularised, fully connected architecture, with ReLU nonlinear activation functions for both the generator and the discriminator. We observed that in practice, using batch normalisation in the generator network harmed performance. A diagram showing the network architecture for the generator network is shown in Figure 4.4. The discriminator network architecture is shown in Figure 4.5. A network diagram, showing the overall training procedure, including both the discriminator and the auxiliary delay discriminator, is shown in Figure 4.6.

4.3.7 Training Parameters

We optimise all networks using an Adam optimiser Kingma and Ba (2017) with a learning rate of 1×10^{-4} . We set the sequence length for the generation and, corresponding, the number of resistance values passed to the discriminator in a single sequence as $s = 10$. We set the sequence length for input to the delay discriminator as $s = 2$ (i.e. a single generation) although, as described in Algorithm 4, the generator is run with either a sequence length of $s = 2$ or a sequence length of $s = 1 + q$ in the case of the delay discriminator, for both the full and split delay settings. These parameters were selected based on empirical performance and observed stability of convergence of the network training.

We run each GAN training experiment for 1000 epochs, with each epoch consisting of 500 steps, where all three networks are updated simultaneously. We set the value of $q_{\max} = 20$. The dimension of the latent prior for the generator, denoted by z , is set equal to 20. We set the

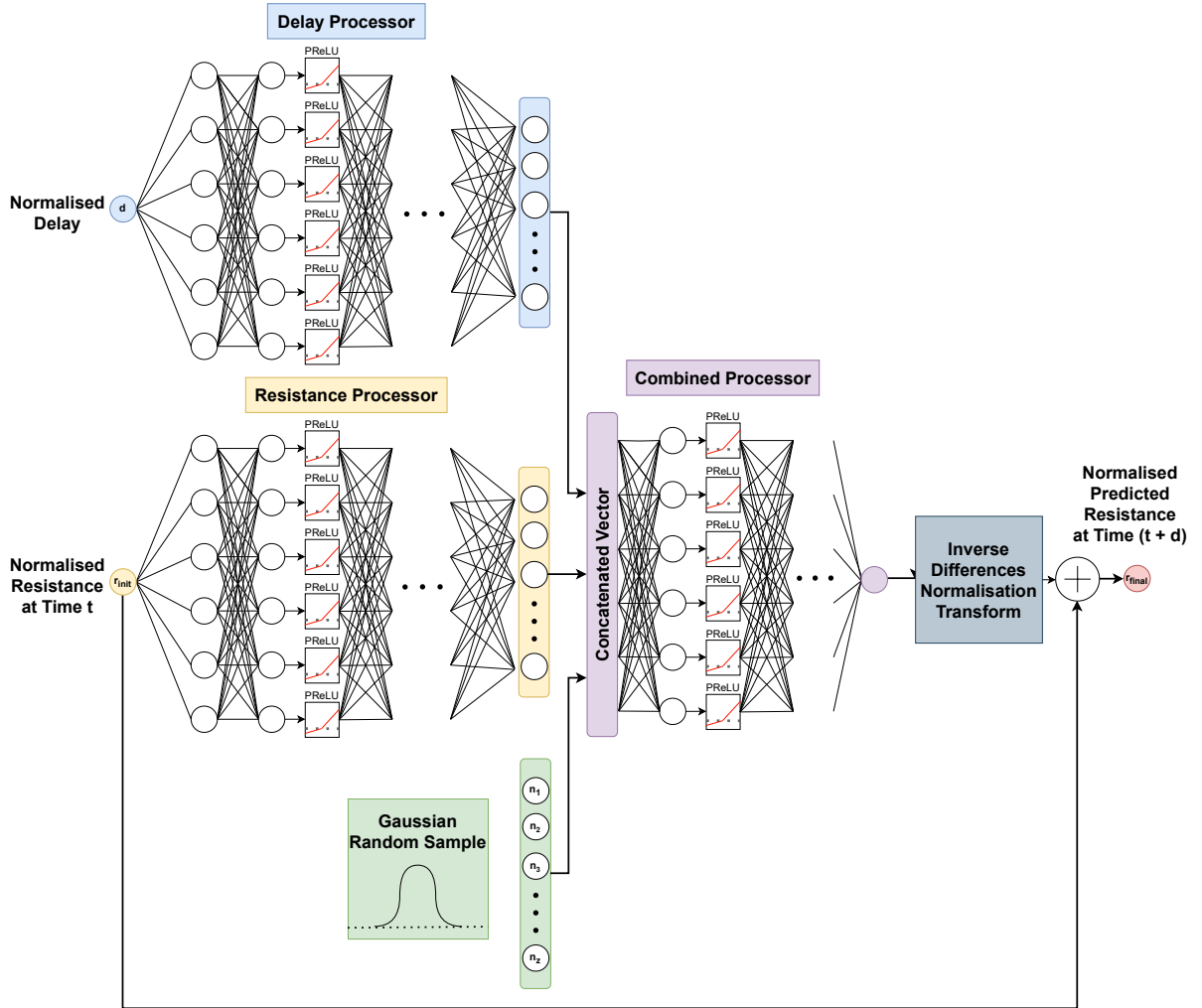


Figure 4.4: Structure of the **cGAN** used for training. The delay processor and resistance processor map the delay d and the normalised resistance at time t , denoted by $\bar{r}_{init}$, to independent embeddings. The combined processor network then processes these embeddings along with the diagonal Gaussian noise vector of dimension z , which is the latent prior for the generative model. The output of the combined processor is a normalised difference, to which the inverse differences transform is applied, followed by addition to $\bar{r}_{init}$ to produce the predicted resistance after delay d , $\bar{r}_{final}$.

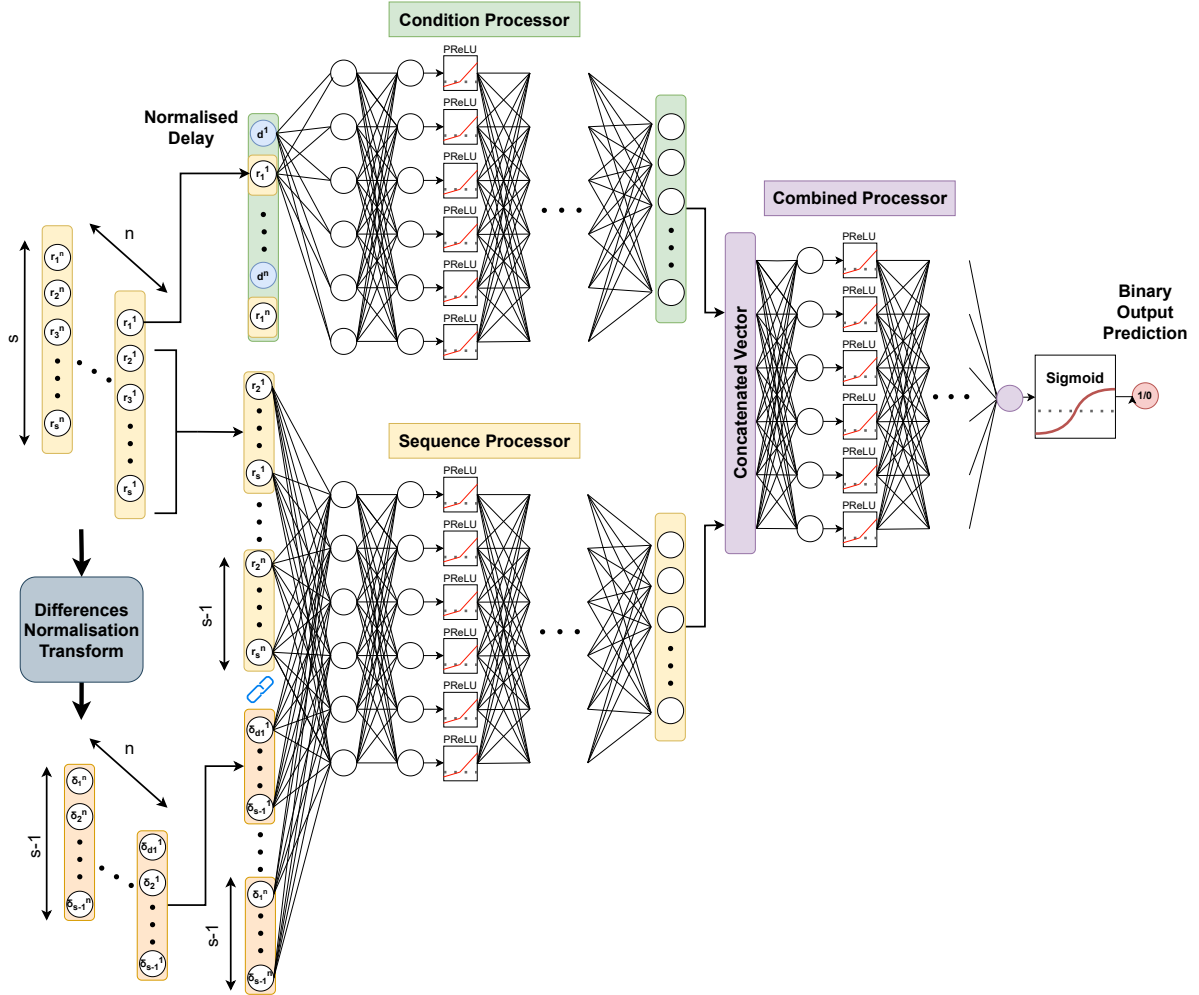


Figure 4.5: Structure of the discriminator architecture and the multiple sample discrimination approach. The condition processor processes the first element of the sequence, which is simply the initial condition, in addition to the delay. The remaining elements of the sequence vector: $r_2 \dots r_s$, are passed to the sequence processor, along with the output of the differences normalisation transform applied to each sequence. The output embeddings of both processors are concatenated and passed to the combined processor. When using the technique of multiple sample discrimination, we pass n sequences (and n delays) to the network simultaneously, with the condition processor and sequence processor input dimensions being scaled up by n and the effective batch size for the discriminator being scaled down by n . The same architecture is used for the main discriminator and the auxiliary delay discriminator described in Section 4.3.5.

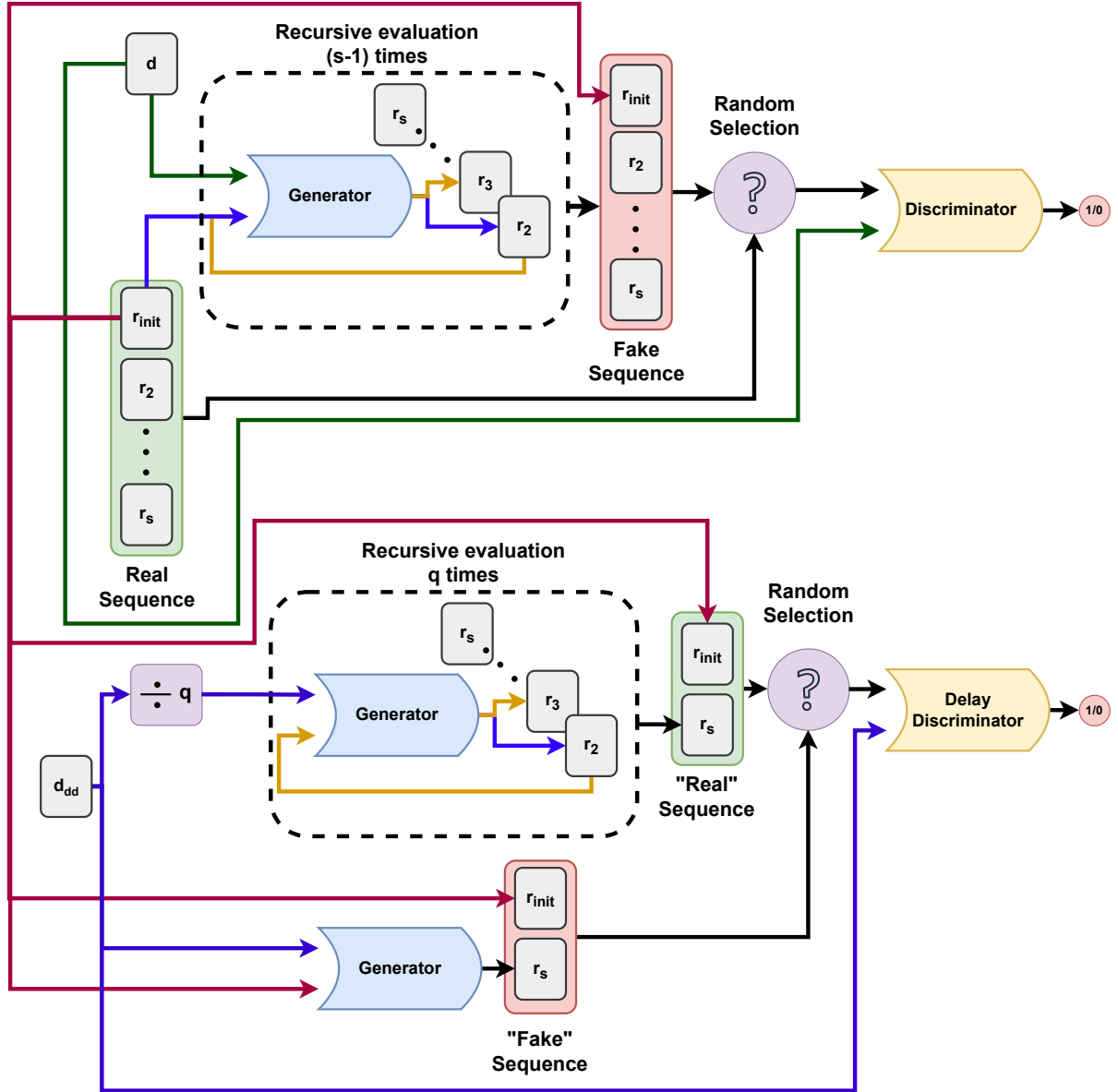


Figure 4.6: The training structure for the generator and the two discriminators. This visualises the delay discrimination procedure. For the purposes of clarity in visualisation, we do not show multiple sample discrimination in this figure. The top of the diagram shows the generation of a fake sequence for the canonical adversarial training, with the discriminator trained to distinguish between real and fake resistance sequences. The bottom of the diagram shows the training of the **cGAN** to enforce consistency in its output for the same total delay, also addressing the problems with closed loop training described in Section 4.3.5.

minimum and maximum delays for the main discriminator as $d_{\min,d} = 1$ and $d_{\max,d} = 90$ and those for the delay discriminator as $d_{\min,dd} = 1$ and $d_{\max,dd} = 500$.

4.4 RESULTS

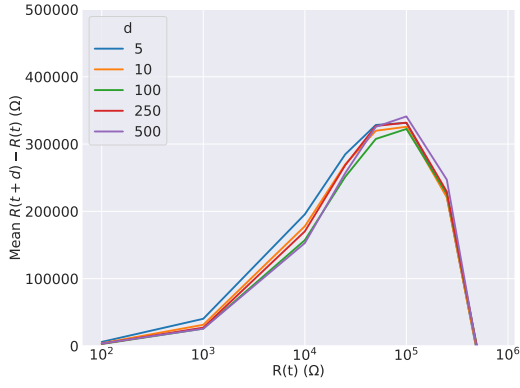
We evaluate the distribution matching performance of the trained generator model, compared to the true distribution of the data in the resistive drift dataset, in several different ways. In order to highlight the consistency of the model output across delays, whose improvement is a key benefit of the delay discriminator approach, we evaluate across a range of equivalent delays using a variety of smaller timesteps. We can also visually compare the series output by the model during closed loop evaluation, which can highlight problems associated with the compounding of closed loop errors and can give some sense of the quality of the generative model. Finally, we compare the statistics of the **cGAN** model output to those of the ground truth event-based model. We statistically estimate the mean and variance (the first and second order statistical moments) of the generated data, as well as showing empirical model outputs for a variety of combinations of delay and initial resistance conditions.

In order to measure the impact of the techniques used to improve the model training, during evaluation, we also perform an ablation experiment which does not make use of the delay discriminator.

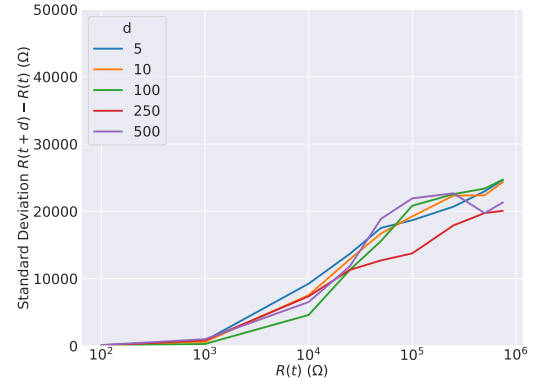
In addition to the evaluation methods in this section, in Chapter 5, we train an autoencoder to store information on memristive devices in the presence of resistive drift noise, making use of the **cGAN** as a channel model. As part of our evaluation of the performance of the autoencoder, in Section 5.4.4, we test the trained autoencoder on the ground truth event-based model. The gap in performance between the test error when evaluated on the **cGAN**, and that when evaluated on the ground-truth model, can be used as a measure of the success of the **cGAN** in matching the statistics of the event-based model for resistance drift.

4.4.1 Delay Consistency Evaluation

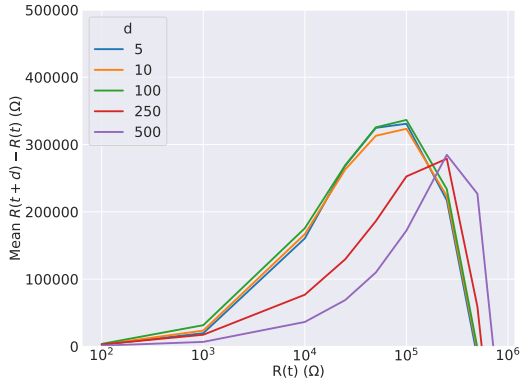
Here, we compare the mean difference from the initial resistance at time $t = 0$, for the model run for an equivalent delay of 500, generated by conditioning with a delay condition of (5, 10, 100, 250, and 500), for (1, 2, 5, 50, and 100) recurrent steps, respectively, in order to achieve the same total delay for the model's final resistance output. We see that the model's output is relatively consistent across delays. We evaluate this behaviour for experiments run with, and without, the delay discriminator. We find that the delay discriminator improves the consistency of the model outputs for the same total delay for different given delay conditions. This is shown in Figure 4.7.



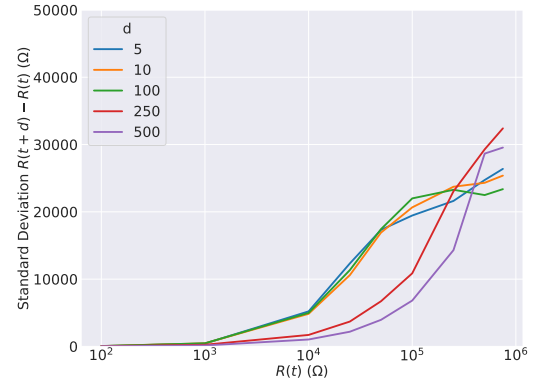
(a) Mean comparison.



(b) Standard deviation comparison.



(c) Mean comparison for delay discriminator ablation experiment.

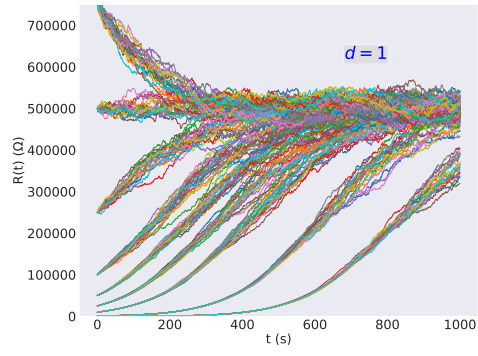


(d) Standard deviation comparison for delay discriminator ablation experiment.

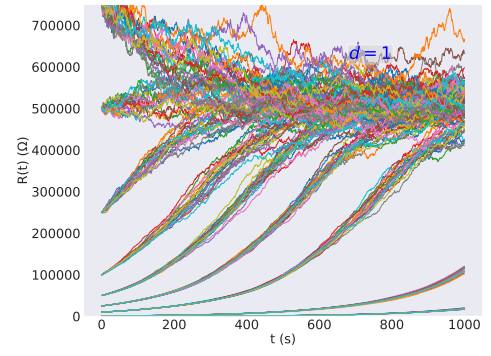
Figure 4.7: Experiments examining the consistency of the model for generation of outputs of the same cumulative delay, ΔT , at different levels of delay refinement d . We measure the mean change in the resistance for a variety of starting resistance values. In the case of a delay of d , we use $500/d$ steps run in order to achieve a total equivalent delay of $\Delta T = 500$.

4.4.2 Qualitative Evaluation

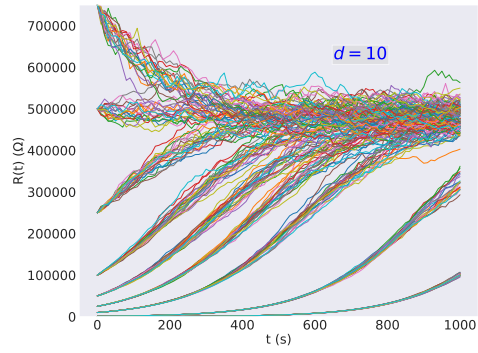
Below, we evaluate the model by recurrently generating time series data for a variety of delay conditions. This is done in order to demonstrate the model's ability to generate visually realistic series for a variety of initial resistance values and conditions, and to exhibit the lack of closed loop errors as part of the recurrent evaluation. In order to evaluate, we generate 20 series for each initial resistance value for a range initial resistances (for any given delay). In each case, we generate for a number of steps equal to $\text{ceil}(1000 + d)/d$ where d is the delay condition. Example series generations are shown for the delay conditions of 1, 10, 100, 500 in Figure 4.8. These can be compared visually to the real series in the dataset, shown in Figure 4.1.



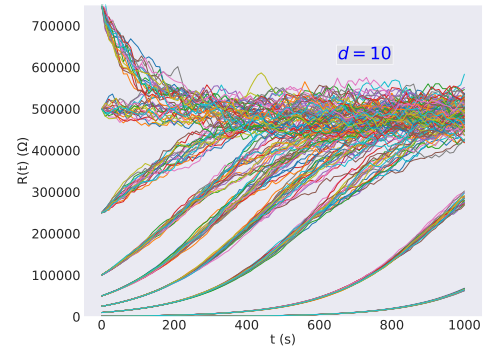
(a)



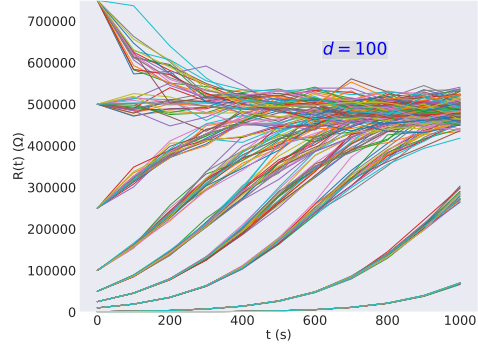
(b)



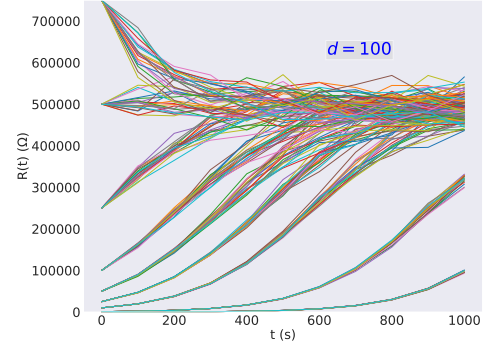
(c)



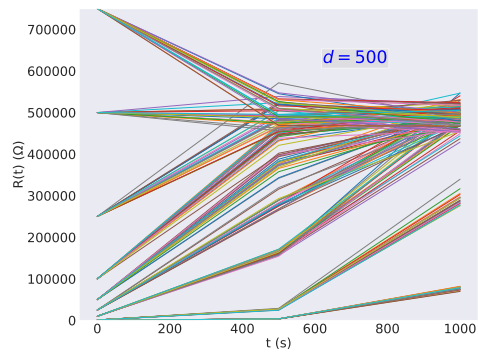
(d)



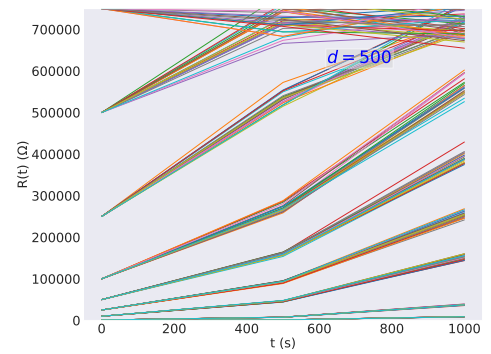
(e)



(f)



(g)

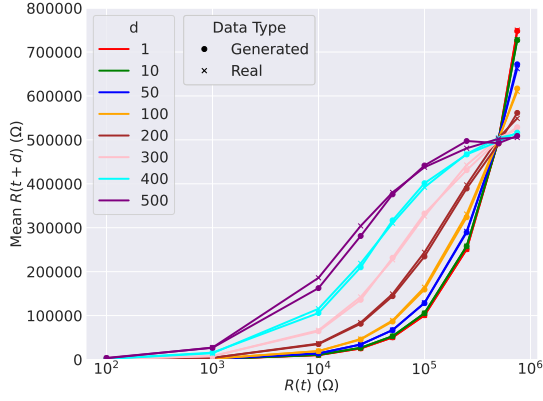


(h)

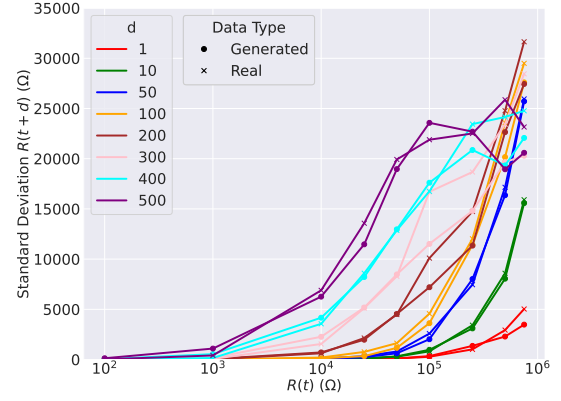
Figure 4.8: Generated series for different delays with (left figures 4.8a, 4.8c, 4.8e, 4.8g) and without (right figures 4.8b, 4.8d, 4.8f, 4.8h) the delay discriminator, for an equivalent delay of 1000, recurrently evaluated for different delay conditions $d \in \{1, 10, 100, 500\}$.

4.4.3 Conditioned Statistics Evaluation

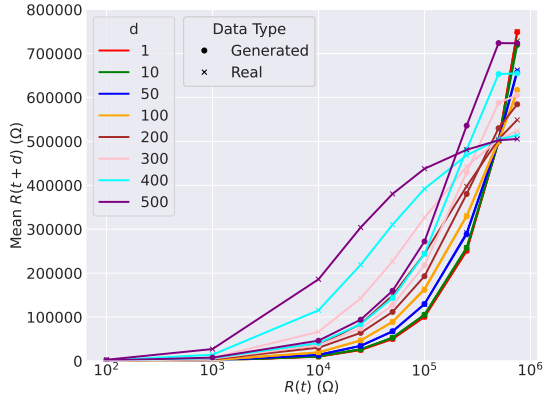
In Figure 4.9 we see a comparison of the statistical means and standard deviations to those obtained using the real model. In order to estimate these statistics, we estimate them using 100 samples of the **cGAN** model output, and do the same for the ground truth event-based model (also using 100 samples per parameter estimate). We plot comparisons for both the proposed **cGAN** model, and for an ablation experiment excluding the delay discriminator.



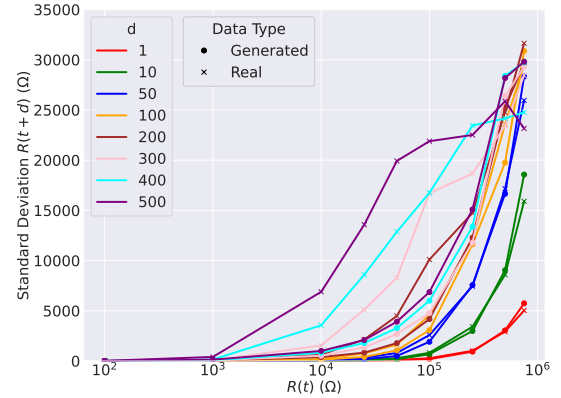
(a) Mean comparison.



(b) Standard deviation comparison.



(c) Mean comparison for delay discriminator ablation experiment.



(d) Standard deviation comparison for delay discriminator ablation experiment.

Figure 4.9: Means and standard deviations of the final resistance conditioned on different combinations of initial resistances and delays, for the generator trained using the delay discriminator (Figure 4.9a and 4.9b) and trained without the delay discriminator as an ablation experiment (Figures 4.9c and 4.9d). The empirical statistics for each condition are plotted alongside those obtained from the ground truth model, in order to demonstrate the ability of the model to fit to the true distribution. We see a marked improvement in the statistical moment matching when using the delay discriminator. Legend entries denote the delay, d that the statistics are evaluated for, with ‘real’ entries being the statistics derived from the ground truth event-based model.

Figure 4.10a shows final resistance values for given initial resistance and delay conditions, simulated using the true event-based model (the ground truth). Figure 4.10b shows a comparison of the final resistance values for the same conditions, with simulations performed by the **cGAN** model. For delays, ΔT , up to $\Delta T = 500$ s, which was the maximum value of d_{ad} (the

delay discriminator delay), we see that the model's performance is accurate. Although, for delays above this (1000s) the **cGAN** model's similarity to the true statistics drops.

Figure 4.10c shows the model output for the same conditions, but using a recurrent evaluation with a delay of $d = \Delta T/10$, for 10 closed loop recursions (feeding the output of the model back to itself as input). We see that recurrently evaluating the model with a smaller delay enables more accurate recreation of the statistics for this larger total delay.

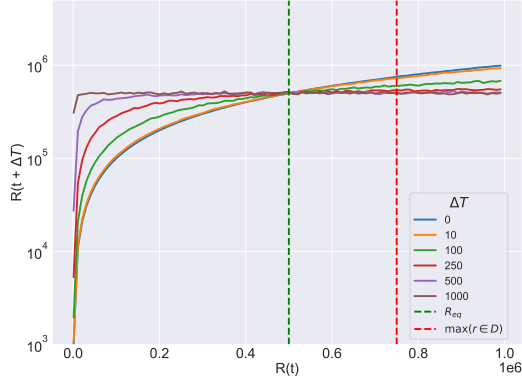
We see, in Figure 4.10d, the model output for the same conditions, for an ablation experiment run without the delay discriminator. As can be seen, the model's accuracy in matching the statistics of the dataset for delays larger than that in the training data (i.e. $d_{\max} = 90\text{s}$) is significantly degraded. When the generator is evaluated recurrently, we see an improved performance, due to the presence of these smaller delay conditions in the training data. The same recurrent evaluation setting for the delay discriminator is shown in Figure 4.10e.

4.5 CONCLUSION

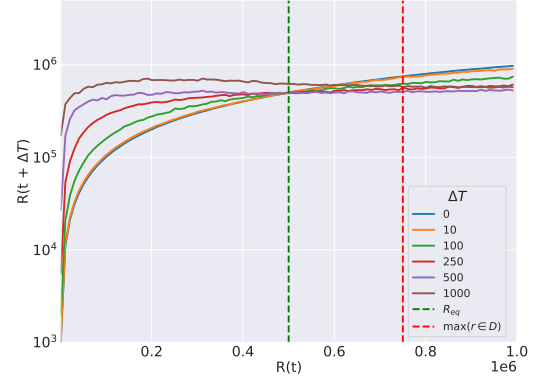
We have seen that in an effort to create a **GAN** that is conditioned directly on the delay condition, we are able to use the technique of delay discrimination to boost the performance and generalisation of the **cGAN** when evaluating at larger delays, improving consistency across the timescales of generation, and enabling the use of the model for a range of delays.

Future work could consider the use of the **cGAN** in simulations of neuromorphic circuits, potentially extending it to a stateful setting to capture memory-dependent phenomena such as the volatility (see Chapter 3). Future work could also include the evaluation of the delay discriminator method more generally, to assess its ability to improving the training performance of generative time series models at a more fundamental level.

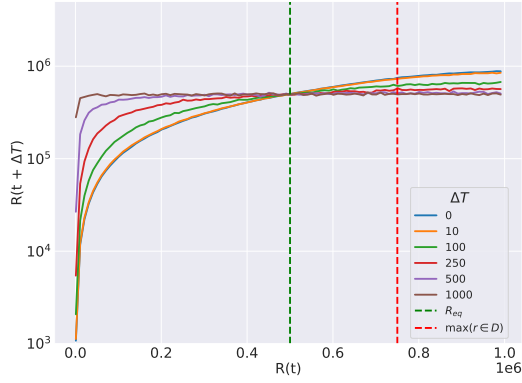
Overall, we have demonstrated that it is possible to effectively model the resistive drift phenomenon in memristors through the use of a **cGAN** trained to model the delay and resistance-conditioned distribution. We saw that we could attempt to tackle multiple challenges associated with the training adversarial networks for time series data, without the use of supervised losses or scheduled sampling, by making use of a the delay discriminator to enforce consistency in the generation of series across a variety of timescales. This technique was also shown to extend the scope of the **cGAN**, for generation of delay-conditioned outputs using a single forward pass, to much larger delays than present in the dataset. Conditioning directly on the delay, rather than resorting to recurrent evaluation, mitigates the number of open-loop recurrent steps that need to be performed to achieve modelling of the conditional distribution for a given delay condition, which will be useful to us when we use the **cGAN** as a channel model for use in the problem of training a deep neural network to store information on memristors in the presence of resistive drift, in the next chapter.



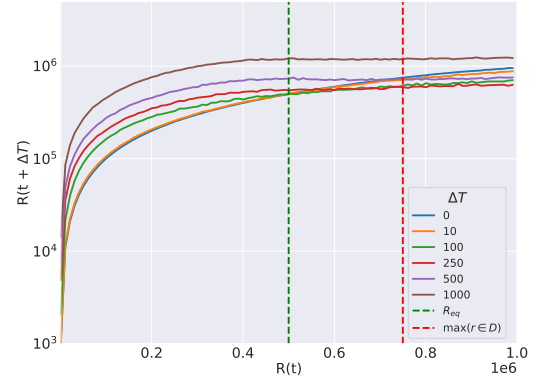
(a) Ground truth model.



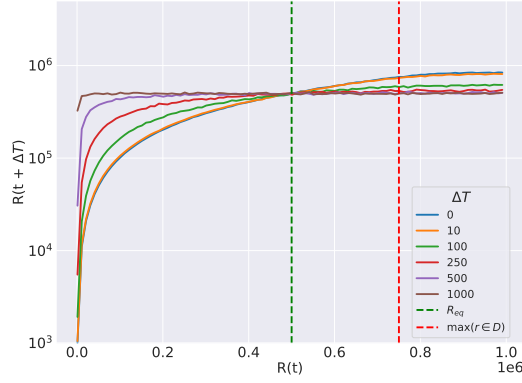
(b) Conditional GAN.



(c) Conditional GAN with recurrent evaluation.



(d) Delay discriminator ablation.



(e) Delay discriminator ablation with recurrent evaluation.

Figure 4.10: Empirically sampled final resistances for given initial and final resistance conditions. Figure 4.10a shows the statistics for the ground truth event-based model. Figure 4.10b shows statistics for the cGAN. Figure 4.10c shows statistics for the cGAN model evaluated recurrently with a delay of $d/10$ and 10 recurrent steps. Figure 4.10d shows an ablation experiment run without the delay discriminator, and Figure 4.10e shows a recurrent evaluation of the delay discriminator ablation experiment with a delay of $d/10$ and 10 recurrent steps.

ENERGY-CONSTRAINED INFORMATION STORAGE ON MEMRISTIVE DEVICES OVER TIME

Information in the natural world is spatio-temporal. The human brain has developed a highly efficient process of dealing with spatio-temporal information in real time, using networks of spiking neurons. Information in the real world is far from static, and is constantly evolving over time. Building upon the differentiable modelling approach we developed in the previous chapter for simulating the phenomenon of resistive drift in memristors, we now turn our attention towards the problem of information storage and recovery in the presence of drift, under energy constraints. We continue our approach of looking at the problem from a communication theoretic perspective, viewing the noisy memristor as a communication channel, and investigate how we can use the technique of Deep Joint Source-Channel Coding ([Deep JSCC](#)) to learn to store information in a data-driven manner. We identify and quantify a key trade-off between the energy expended in programming a device and the reliability of the programmed states over time, modelling each state to have a stochastic lifetime until it degrades to a non-informative equilibrium state. We use this trade-off to study the possibilities for energy-aware information storage on memristors, specifically optimising the end-to-end storage and recovery of natural images, taking into account prior knowledge of the time at which the information will be recovered, presenting techniques for introducing delay-conditioning to a [Deep JSCC](#) autoencoder. Our study of energy-constrained information storage makes steps towards a clearer understanding of the limits of compression and lossy storage of information over time on memristive devices.

5.1 INTRODUCTION

Information storage on memristive devices is a fundamental challenge that needs to be overcome if the devices are to be used in a neuromorphic computing setting. Information storage can be performed in a variety of ways: memristive devices can be considered binary storage devices, multilevel storage devices, or continuous analogue storage devices. The characterisation of optimal methods for storage of information on memristors, exploiting as much of their device capacity as possible, taking into account the non-idealities present, is an important and fundamental challenge.

A range of different kinds of noise affect memristive devices. As noted in Section [2.2.3](#) of Chapter [2](#), these types of noise fall into several different categories; namely: programming noise, sneak-path current noise, read noise such as $\frac{1}{f}$ (pink) noise, device parameter heterogeneity due to manufacturing differences, and finally resistive drift or other noises associated with the metastability of the resistive state such as burst noise (random telegraph noise). All but one of these types of noise are spatial noises - they are time-invariant. Resistive drift noise is distinct in that it is a type of noise dependent on the delay between writing a value to

the device and subsequently attempting to recover it accurately. In this chapter we focus on resistive drift in the context of energy-constrained storage. We explore a fundamental trade-off between an energy budget and the capacity of a device for storage, and use [Deep JSCC](#) to learn to maximise the use of the available capacity of the devices for the end-to-end storage of information. As noted in [Chapter 3](#), drift is of particular interest from a neuromorphic perspective due to its similarity to synaptic weight decay in neurons, or membrane potential leakage for unstable memristors. It is therefore important to study this type of noise and to characterise the limits of the information capacity of memristive devices in the presence of drift. We will explore techniques for adaptive storage of information, with knowledge of the expected time of recovery, according to energy constraints.

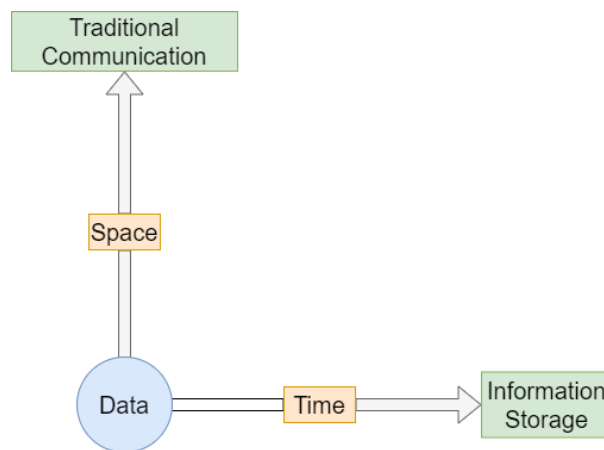


Figure 5.1: Communication of information occurs through both space and time. In traditional wireless communication, the majority of the challenge is associated with transmission and recovery of information through space, whereas in the case of storage, we are more concerned with how the passage of time affects the quality of the recovered information. In both cases, distortions are introduced by the conditional channel distribution, for a given source alphabet.

5.1.1 Information Storage as a Communication Problem

Information storage can be seen as a problem of communication. In contrast to the possibly more familiar communication setting - the challenge of transmitting information from one *spatial* location to another - information storage is the problem of transmitting information from one *temporal* point to another (see [Figure 5.1](#)). Motivated by this communication-theoretic view of data storage, we formulate our problem as one of the transmission of information over a noisy channel, and consider the use of deep learning techniques from the field of communications for addressing this problem of storage in the presence of resistive drift. A communication channel is simply a conditional probability distribution for a given source alphabet, and the resistive drift is such a distribution (parameterised by the delay). The treatment of resistive drift as a problem of communication is formalised in [Section 5.2](#).

The mutual information between the stored resistance and the output resistance decreases with increasing delay. In addition, the choice of possible initial resistances has a strong impact on this mutual information, with a smaller number of possible values decreasing the mutual information. There is firstly a trade-off to be made between the spread of initial resistance

conditions and the information rate of our coding approach (in general, the lower the initial resistance, the more deterministic the transformation introduced by the storage channel). We must simultaneously consider the impact of energy consumption on the storage problem. The energy-storage trade-off has thus far, to the best of our knowledge, not been considered explicitly in this context. However, a number of works have studied the energy requirements of bringing memristors into particular resistive states [Fu et al. \(2020\)](#); [Ielmini \(2011\)](#); [Ielmini et al. \(2010\)](#); [Wei et al. \(2011\)](#).

In Section 5.3, we introduce and characterise the fundamental trade-off between energy consumption and information storage lifetime in memristive devices, demonstrating how this presents itself as an optimisation problem to be solved. In order to optimise this problem according to the proposed energy-information trade-off, we make use of a [Deep JSCC](#) autoencoder architecture - a deep learning based approach to the end-to-end communication of information over a noisy channel, involving the optimisation of a particular distortion metric as the objective. The approach requires the simultaneous consideration of a representation learning problem (source coding) for a dataset, and the problem of recovery of the distorted representations following their transmission over a noisy channel (channel coding).

Previous approaches to the problem of coding against the noise and stochasticity involved in the process of storing information on certain memristive devices for the purposes of information storage were discussed in Section 2.4.2 of Chapter 2. A [Deep JSCC](#) approach to coding against programming noise in Phase Change Memory ([PCM](#)) devices has been previously developed [Zarcone et al. \(2018\)](#), which we use as a basis to address the problem of resistive drift. The authors made use of a differentiable Gaussian kernel model that was fit to a programming noise dataset, and trained an autoencoder to encode information in the presence of this noise. We adopt a similar approach to information storage in this chapter, additionally making use of delay conditioning for the [Deep JSCC](#) autoencoder, through introducing a modified form of Generalised Divisive Normalisation ([GDN](#)) that we call Conditional Gaussian Divisive Normalisation ([CGDN](#)), as well as the use of fully connected residual networks, to incorporate information about the delay into either the encoder, the decoder, or both networks simultaneously. This conditioning approach could also potentially be applied more generally to other channel characteristics in different settings, such as the Signal to Noise Ratio ([SNR](#)).

5.2 PROBLEM FORMULATION

In order to use the differentiable channel model we developed in Chapter 4 to learn to store information on a set of memristors, we use a technique known as [Deep JSCC](#). This involves the use of an autoencoder (a kind of deep neural network) to minimise the distortion of information that is compressed and then decompressed after passing the compressed representation of the information through a differentiable model of a communication channel. This has been shown to be an effective approach to Joint Source-Channel Coding ([JSCC](#)) that is able to minimise the distortion subject to a particular information rate for Additive White Gaussian Noise ([AWGN](#)) channels with specific [SNR](#) values [Bourtsoulatze et al. \(2019\)](#) and even for classes of channels parameterised by their [SNR](#) [Xu et al. \(2021\)](#) through an architectural modification using an

attention mechanism. In this setting, perfect recovery is no longer possible or necessary (or even particularly desirable¹), and a trade-off can be made between the rate of information storage (for example, the number of devices used for storing a particular piece of information) and the chosen distortion upon recovery, e.g. a Mean Squared Error (MSE) for images.

The use of Deep JSCC to transmit information over noisy communication channels has grown as a field in recent years. Traditional separate source and channel coding techniques were necessary, despite their lack of optimality in shorter block-length regimes, due to the unfeasible overhead of designing specialised joint coding schemes for every combination of channel and source. As a result, the seemingly good trade-off provided by the modular design of separate source and channel coders between simplicity and effectiveness meant that little attention was given to designing joint schemes. This trend has changed with the advent of deep learning, which provides a method for automation of intricate and time-consuming tasks which would formerly need to have been performed by hand. Deep JSCC is the use of deep neural networks to perform joint source and channel coding of particular kinds of sources, such as images, over specific kinds of communication channel Bourtsoulatze et al. (2019).

Work has also been done in examining how a single Deep JSCC network can be used to transmit a source over a collection of parameterised channels. The SNR - a channel characteristic that can be used to parametrise AWGN channels - has been incorporated into Deep JSCC Autoencoders as a condition through the use of attention mechanisms Xu et al. (2021).

We extend the notion of conditioning the network on the channel characteristics, by introducing alternative delay conditioning techniques to the Deep JSCC Autoencoder. We consider that the condition need not be SNR, but could be any continuously varying channel characteristic. In the memristor setting, we can therefore parametrise the models by the delay of the memristor channel model.

We consider the storage of one particular kind of information source - images - on memristive devices that are subject to resistive drift. We wish to design a single Deep JSCC network that is capable of storing information on an array of such devices, conditioned upon the delay, d .

Let us denote a single image $X \in \mathbf{R}^{H \times W \times C}$. Our goal is to perform storage of X on an array of m memristor devices, for a random delay distributed according to a distribution $p_d(d)$ with support $[d_{\min}, d_{\max}]$, such that we minimise the expected value of the distortion, l , (the MSE between the image and its reconstruction, $\hat{X} \in \mathbf{R}^{H \times W \times C}$). Let $X_{h,w,c}$ be an indexing of the three-dimensional tensor X that represents our image, then the distortion, l , is defined as:

$$l(X) = \frac{1}{H \cdot W \cdot C} \sum_{h,w,c} (X_{h,w,c} - \hat{X}_{h,w,c})^2 \quad (5.1)$$

The reconstructed random variable $\hat{X}$ is the output of a parameterised encoder function, E_{θ_e} , applied to the image before storage; the random variable distributed according to the stochastic channel conditional distribution conditioned on a delay d for a given initial resistance r , $\hat{R}_{r,d}$;

¹ For certain types of information which is intended for human consumption rather than logical processing. For example, digital images are often lossily compressed using codecs such as JPEG, whereas lossless compression should always be used for data which is not tolerant of distortions, such as the bits defining a computer program.

and a parameterised decoder function D_{θ_d} applied to the output of this channel. It is thus a random variable that is given as follows:

$$\hat{X} = D_{\theta_d}(\hat{R}_{d, E_{\theta_e}}(X)) \quad (5.2)$$

The problem is then one of minimising the expected distortion, given the distribution $p_d(d)$, by choosing an appropriate function E_{θ_e} and D_{θ_d} from classes of functions parameterised by the parameters θ_e and θ_d , respectively. Note that in our setting, these parameters will be deep neural network parameters, but in general, they could be the parameters for any class of functions, which is the reason for our using this general notation. Thus, the optimisation problem is characterised finding the optimal joint parameters θ_e^* and θ_d^* that satisfy the following:

$$\{\theta_e^*, \theta_d^*\} = \arg \min_{\theta_e, \theta_d} \mathbb{E}_{p_d(d), p_x(x)} [l(X)] \quad (5.3)$$

where $p_x(x)$ is the distribution from which the images X are drawn.

The set of acceptable channel input distributions, induced by $E_{\theta_e}(X)$, can be limited based on energy constraints, thus imposing constraints on the parameters θ_e . We characterise the trade-off between the energy consumption and the communication theoretic capacity of the channel, by introducing the energy cost function in Section 5.2.2, used to constrain the set of input distributions for the channel and implicitly impose an energy-constrained channel capacity (upper bound on the achievable rate of communication over the channel). Over the various delay conditions in the support of $p_d(d)$, the information storage problem presents a family of communication channels, with different channel capacities. Of course we might ask the question - why should we not simply choose the capacity achieving input distribution in every case? However, we must also consider the energy cost associated with a particular channel input distribution. This translates to a trade-off between the energy we use to store information and the *energy-constrained* capacity of the devices, with energy consumption as detailed in Section 5.2.2, and to improve rate of information transmission across the channel we must therefore increase the energy used to store information.

For handcrafted codes, it would be complex to engineer any solution other than coding for a few specific choices of storage device channel capacity. We will see that by using deep learning, we can overcome this limitation and create highly specialised codes for storing information on devices, by adapting the input scheme to the particular channel characteristics (Section 5.3). The delay to recovery in the setting of storage in the presence of resistive drift is analogous to the noise power in the more common spatial communication setting in e.g. an [AWGN](#) channel, where the transmission power can be altered to improve the [SNR](#) of individual channels. Here, we do the same for memristive drift channels, by using different input distributions with different associated energy costs, in a delay-adaptive way.

5.2.1 Capacity of the Storage Channel

Given the delay conditioned storage distribution $P(\hat{R}|R, d)$, where $\hat{R}$ denotes the output of the delay channel for a given delay d and initial state R , we can characterise the information content of the storage channel by finding the capacity achieving input random variable, $R^* \in \mathcal{R}$, which maximises the mutual information between $\hat{R}$ and R and achieves thus achieves the capacity of the channel. We denote the capacity by C . Thus:

$$C(d) = \max_{R \in \mathcal{R}} I_{P(\hat{R}|R, d)}(R; \hat{R}) \quad (5.4)$$

Note that usually, the set $\mathcal{R}$, to which the capacity achieving input distribution belongs, is unconstrained. If, however, we have an average power (energy) budget for programming, we must modify the optimisation to obtain the energy-constrained capacity, denoted by C_E , by examining the maximisation problem over the set of valid input distributions - distributions which satisfy the energy budget condition, for a budget B , given as:

$$E[\mathcal{E}(\hat{R})] \leq B \quad (5.5)$$

where $\mathcal{E}(\cdot)$ is the energy equation, mapping a given input codeword (resistance, or state) $\hat{R} = \hat{r}$ to an associated energy cost.

We can denote the set of input distributions for which the condition B is satisfied as $\mathcal{R}_B \in \mathcal{R}$, thus the energy-constrained capacity (for the reduced set of possible input distributions) is given as follows:

$$C(d) = \max_{R \in \mathcal{R}_B} I_{P(\hat{R}|R, d)}(R; \hat{R}) \quad (5.6)$$

We note that for short-term storage, it may be acceptable to store information across a small range of states, since the channel capacity is high. In the case of longer-term storage, the channel capacity for a given energy budget decreases significantly, and a much larger amount of energy must thus be expended to increase the channel capacity. The approach of using [JSCC](#) for information storage for different delays (timescales) is thus expected to vary. The approach taken when jointly optimising across timescales may also vary.

In the next section, we derive a parameterised function $\mathcal{E}$, that defines the energy consumption for a given channel input distribution in the context of the problem of resistance drift in memristors.

5.2.2 Energy-Retention Trade-off

A trade-off has been noted in the memristor literature between the total energy of a pulse needed for programming a device and the subsequent dynamic range or retention time. This

has been attributed to a variety of phenomena, notably the thickening of a conductive filament. When a pulse of extended duration is applied to the devices, the final width of a formed conductive filament increases.

If the dynamic range of the device (which is characterised by the lowest available state for storage) is decreased, for a given energy budget, this decreases the information capacity of the device, since the capacity of the storage channel - the delay conditioned distribution - is a function of the number of states available for storage. Simultaneously, the energy budget constrains the allowable input distributions, resulting in an upper bound on the achievable rate that is also dependent on the energy.

As, has been noted in Section 5.1, a number of works have identified a relationship between the energy expenditure in bringing a memristor into a particular resistive state, and the subsequent retention time or stability of that state. For example, filament size in tantalum oxide thin film memristors has been observed to impact the magnitude of the high and low resistive states [Wei et al. \(2011\)](#).

Fu et al. reported a linear relationship between applied pulse width and the mean retention time for bio-voltage memristors (Electrochemical Metallization (ECM) devices catalysed by protein nanowires) [Fu et al. \(2020\)](#). Ielmini examined the retention time as a function of the size of resistive filaments for NiO thin-film memristive devices, giving the final size of the resistive filaments as a function of the input signal pulse width applied to the devices [Ielmini et al. \(2010\)](#). Further to this, pulsing experiments performed on H_fO_x Metal-Insulator-Metal (MIM) memristive devices were used to derive a generic inverse proportionality relationship between the compliance current and the final resistance of the memristive state achieved [Ielmini \(2011\)](#).

Based on these findings, we have derived a generic relationship between the initial resistance (assumed to be the off or RESET state) of the devices and the energy needed to bring them into a particular state from this state. This is taken into account in training an autoencoder for storage, through the introduction of a regularisation factor in the loss function to represent the energy trade-off.

5.2.2.1 Derivation of the Energy Trade-off

When programming a memristive device, we can modulate either the current pulse width, or its magnitude, to modify the final switching state. As was explained above, it has been observed that for various kinds of memristive device:

1. **For a fixed current magnitude:** a proportional relationship between the final conductance state of metal oxide memristive devices and the width of current pulses used to program the devices.
2. **For a fixed pulse width:** An inversely proportional relationship exists between the compliance current magnitude for a fixed pulse width and the final resistance state.

We use these two principles in the following discussion to derive a generic trade-off between the energy applied to the device, through the application of a current pulse of a given magnitude and duration, and the final resistive state achieved.

We will make use of the simplifying approximation that the amplitude of the current in an applied signal used to program the devices is constant and equal to its maximum (compliance) value throughout.

We begin by defining the relationship between a given compliance current magnitude and the final resistance attained. Let G_{init} and G_{final} be the initial and final conductance states, respectively, and R_{init} and R_{final} be the corresponding initial and final resistance states. According to our assumption of the proportionality of the final conductance to the compliance current magnitude, for a particular compliance current i_{max} :

$$G_{\text{final}} \propto i_{\text{max}} \quad (5.7)$$

Given that the total change in the conductance over the duration of the pulse is equal to $(G_{\text{final}} - G_{\text{init}})$, based on our assumption that the change in the conductance is linearly proportional to the length of an applied current pulse, we can say that the proportion of this change effected is linearly proportional to the time τ elapsed since the start of the pulse (a first order approximation). Let τ_{final} be the (fixed) length of the pulse, required to bring the resistance to R_{final} (or the conductance to G_{final} , equivalently) from a starting resistance R_{init} (conductance G_{init}). Let R (G) be the resistance (conductance) at time τ during the pulse. Note that $R > 0$. Then, we can express the relationship between τ and R as follows:

$$\begin{aligned} G &= G_{\text{init}} + \frac{\tau}{\tau_{\text{final}}} \cdot (G_{\text{final}} - G_{\text{init}}) \\ \frac{1}{R} &= \frac{1}{R_{\text{init}}} + \frac{\tau}{\tau_{\text{final}}} \left(\frac{1}{R_{\text{final}}} - \frac{1}{R_{\text{init}}} \right) \\ R &= \frac{1}{\frac{1}{R_{\text{init}}} + \frac{\tau}{\tau_{\text{final}}} \left(\frac{1}{R_{\text{final}}} - \frac{1}{R_{\text{init}}} \right)} \end{aligned} \quad (5.8)$$

$$\begin{aligned} R &= \frac{R_{\text{init}}}{\left(1 - \frac{\tau}{\tau_{\text{final}}} + \frac{\tau}{\tau_{\text{final}}} \cdot \frac{R_{\text{init}}}{R_{\text{final}}} \right)} \\ \tau &= \tau_{\text{final}} \cdot \frac{\frac{R_{\text{init}}}{R} - 1}{\frac{R_{\text{init}}}{R_{\text{final}}} - 1} = \tau_{\text{final}} \cdot \frac{\frac{1}{R} - \frac{1}{R_{\text{init}}}}{\frac{1}{R_{\text{final}}} - \frac{1}{R_{\text{init}}}} \end{aligned} \quad (5.9)$$

We see in Equation 5.8 the resistance expressed as a function of time. The inverse relationship (time as a function of resistance) is given in Equation 5.9. Using these equations, it can be verified that $t = 0$ gives $R = R_{\text{init}}$ and $t = \tau_{\text{final}}$ gives $R = R_{\text{final}}$.

In order to obtain the amount of energy necessary to bring the resistance to R_{final} , we must integrate the power with respect to time. Note that power is equal to $i^2 R$ for a constant resistance and current. Since for a memristor the resistance changes as a function of time, we must take this into account in the integral.

As noted above, the maximum (compliance) current is approximately inversely proportional to the final resistance obtained after the full current pulse duration, τ_{final} .

$$i_{\max} = \frac{K}{R_{\text{final}}} \quad (5.10)$$

In order to satisfy the boundary condition that $i_{\max} = 0$ for $R_{\text{final}} = R_{\text{init}}$, we have:

$$i_{\max} = K \cdot \left(\frac{1}{R_{\text{final}}} - \frac{1}{R_{\text{init}}} \right) \quad (5.11)$$

We will make the simplifying approximation that the current is always equal to its maximum compliance value during a current pulse (i.e. $i(t) = i_{\max}$). We can use Equation 5.8 and Equation 5.11 to integrate the power (as a function of time) to obtain a relationship between the final resistance value and the energy expenditure, $\mathcal{E}$, for a constant current and a final resistance value that is assumed inversely proportional to the length of a current pulse of a fixed magnitude. Let us assume that the energy expenditure is symmetric, thus the amount of energy required to change the state from $R_1 \rightarrow R_2$ is equal to that required to change from $R_2 \rightarrow R_1$, thus taking the absolute value of the integral to obtain the energy expenditure:

$$\begin{aligned} \text{Let } J &:= \frac{1}{\tau_{\text{final}}} \left(\frac{1}{R_{\text{final}}} - \frac{1}{R_{\text{init}}} \right) \\ \mathcal{E}(R) &= \left| \int_{t=0}^{t=\tau} i(t)^2 R(t) dt \right| \\ &= \left| \int_{t=0}^{t=\tau} \left(K \left(\frac{1}{R_{\text{final}}} - \frac{1}{R_{\text{init}}} \right) \right)^2 \frac{1}{\frac{1}{R_{\text{init}}} + tJ} dt \right| \\ &= \left| \left[\left(K \left(\frac{1}{R_{\text{final}}} - \frac{1}{R_{\text{init}}} \right) \right)^2 \frac{\ln \left| \frac{1}{R_{\text{init}}} + tJ \right|}{J} \right]_0^{\tau} \right| \\ &= \left| \frac{\left(K \left(\frac{1}{R_{\text{final}}} - \frac{1}{R_{\text{init}}} \right) \right)^2}{J} \ln \left| \left(\frac{1}{R_{\text{init}}} + \tau J \right) R_{\text{init}} \right| \right| \\ &= \left| \tau_{\text{final}} K^2 \left(\frac{1}{R_{\text{final}}} - \frac{1}{R_{\text{init}}} \right) \ln \left| \left(\frac{1}{R_{\text{init}}} + \tau J \right) R_{\text{init}} \right| \right| \end{aligned}$$

Substituting Equation 5.9:

$$\mathcal{E}(R) = \left| \tau_{\text{final}} \cdot K^2 \cdot \left(\frac{1}{R_{\text{final}}} - \frac{1}{R_{\text{init}}} \right) \cdot \ln \left| \frac{R_{\text{init}}}{R} \right| \right| \quad (5.12)$$

$$= \left| A \cdot \ln \left| \frac{B}{R} \right| \right| \quad (5.13)$$

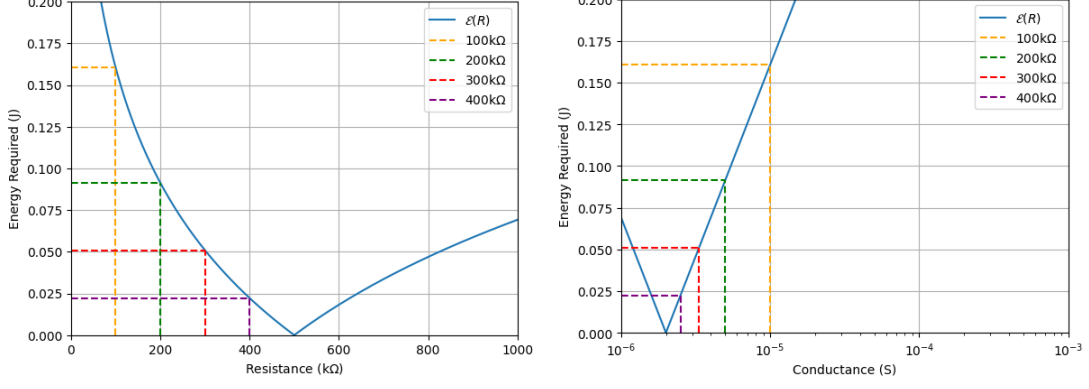


Figure 5.2: Costs associated with bringing the resistance of a memristor to the specified value, starting from $R_{\text{init}} = 500\text{k}\Omega$, $\tau_{\text{final}} = 1.0$, $K = 10.0$, and $R_{\text{final}} = 1\text{k}\Omega$. We see that the energy costs marginally increase as the desired final resistance is lowered, whereas they marginally increase as the desired final resistance is increased beyond R_{init} . The right hand plot shows the trade-off plotted in terms of the resistance, whereas the left hand plot shows the trade-off in terms of the conductance, on a logarithmic scale, where the trade-off is linear, demonstrating the exponentially increasing marginal costs of bringing the device into a lower resistive state.

Equation 5.12 has a choice of constant parameters $A \triangleq \tau_{\text{final}} \cdot K^2 \cdot \left(\frac{1}{R_{\text{final}}} - \frac{1}{R_{\text{init}}} \right)$ and $B \triangleq R_{\text{init}}$, that scale and determine the shape of the energy cost as a function of R , respectively. In the next section, this energy cost function will be used as a regularisation term in the training of an autoencoder to store information as resistance values on memristors, resulting in a trade-off between the range of states available for storage (with those at lower resistance levels being more stable²) and the average energy consumption required to bring the device down to that resistance level. In our experiments, we set R_{init} equal to the equilibrium resistance, since this resistance can be achieved without expending any energy, through the autonomous drift of the resistance of the device, in the absence of a programming pulse.

The energy cost is shown in Figure 5.2, for $\tau_{\text{final}} = 1.0$, $K = 10.0$, $R_{\text{init}} = 500\text{k}\Omega$, and $R_{\text{final}} = 1\Omega$. Note that in Section 5.3, in Equation 5.18, we make use of the energy budget Equation 5.12, using a single scaling factor $\lambda_{\text{energy}} = K^2/\tau_{\text{final}}$ to linearly scale the absolute consumption, maintaining the shape of the energy cost function.

5.3 METHOD

Here we detail the specifics of the autoencoder network training procedure. We make use of the conditional Generative Adversarial Network (cGAN) channel model developed in Chapter 4 to simulate the resistive drift in a differentiable and computationally efficient manner, modelling the delay and initial resistance conditioned resistive drift distribution.

² Lower resistance states can be seen as channels that have a higher “SNR”, due to the inverse relationship between the variance of the final resistance output for a given delay and the initial resistance value.

5.3.1 Data Normalisation

As was necessary for the training of the **cGAN** model in Chapter 4, data normalisation forms an essential initial step for ensuring our training data is presented to the models in a suitable form.

In the case of the resistance values, we make use of the same resistance normalisation transform given in Equation 4.6 of Chapter 4. We do not explicitly specify this transform. Rather, we assume the output of the encoder to be in the normalised space, and pass the outputs of E_{θ_e} - the encoder - directly to the generator model of the **cGAN** channel model, which operates in this normalised space. Similarly, the input to D_{θ_d} - the decoder - is assumed to be in the normalised resistance space. The data normalisation transform for the resistance values was given in Equation 4.6 of Chapter 4.

In the autoencoder, the goal is to transform the input distribution contingent upon the delay, and there may be a highly nonlinear relationship between the input delay and the associated transformation. Thus, we apply a logarithmic transformation to the delay before passing it to the encoder and/or decoder networks, as shown in Equation 5.14. In this transform, we add a small constant to the delay before taking the logarithm, to lower bound the output and prevent it from reaching $-\infty$. Note that in the case of the autoencoder, only positive delays are considered, thus $d \geq 0$, ensuring a valid domain. The transformation is dependent on the delays we use for training, which are uniformly distributed integers in the range $[d_{\min}, d_{\max}]$. In the case that $d_{\min} = d_{\max}$, this constitutes training for a single delay condition³ and we set $\mu_d = 0$ and $\sigma_d = 1$, otherwise, we compute the delay normalisation transform for the autoencoder according to:

$$\begin{aligned}\mu_D &= \frac{1}{d_{\max} - d_{\min}} \sum_{d \in [d_{\min}, d_{\max}]} \ln(d + 0.1) \\ \sigma_D &= \sqrt{\frac{1}{d_{\max} - d_{\min}} \sum_{d \in [d_{\min}, d_{\max}]} (\ln(d + 0.1) - \mu_D)^2} \\ \bar{d} &= \frac{\ln(d + 0.1) - \mu_D}{\sigma_D}\end{aligned}\tag{5.14}$$

A histogram of the transformed delays in the dataset (delays in the range $[1, 1000]$) is visualised in Figure 5.3.

5.3.2 Regularisation

We introduce regularisation terms to satisfy the energy constraints based on the derived energy trade-off for a particular input distribution (storage scheme), as well as constraints on the range of allowable encoded resistance values. We use two forms of regularisation for training our network, transforming the problem from simple optimisation of the **MSE** of image

³ This scenario never arises in delay-conditioned experiments, since if there is only one delay condition, then conditioning on the delay is redundant, however we include this point for the sake of completeness.

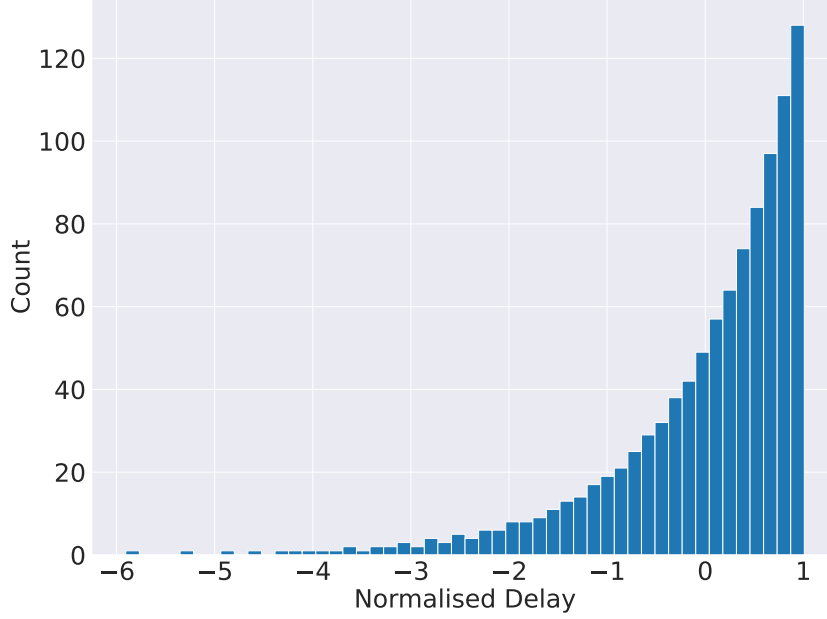


Figure 5.3: Histogram of the delays following the logarithmic transformation used for training and conditioning the autoencoder.

reconstruction into one where a trade-off must be made between three loss terms: the [MSE](#), a resistance regularisation term, and an energy regularisation term.

5.3.2.1 Resistance Regularisation

The first regularisation term is used to ensure that the resistances output by the encoder network fall into a valid range for the dataset (note that the logarithmic transform applied to the resistances, as given in Equation 4.6, means that we take an exponential transform (the inverse) at the output of the encoder, ensuring that the encoded resistance values are always positive).

In order to enforce these limits, we use a squared error loss when the values of the encoder output violate the boundaries [Zarcone et al. \(2018\)](#). We make use of hard regularisation limits in addition to the soft limits, to prevent pathological encoder outputs early on during the training process, to avoid the problem of gradient explosion, where excessively large gradient values during optimisation may cause the model training to fail early on.

We denote the upper, soft regularisation limit by $R_{\text{high}} \leq \frac{1}{g_{\text{parallel}}} = (5 \times 10^5)$. Similarly, we denote the lower, soft regularisation limit by $R_{\text{low}} \geq 0$. The hard upper and lower limits are denoted b_{high} and b_{low} respectively, and set to be equal to the maximum resistance values we want our network to be able to generate. The soft limits are enforced through regularisation, whereas the hard limits are enforced through non-differentiable encoder output clipping at the given values. We compute the soft regularisation loss in the transformed logarithmic space, to allow for fine-grained control that is less dependent on order of magnitude.

The upper soft resistance regularisation term, $r_{\text{upper}}(\cdot)$, is given as:

$$r_{\text{upper}}(X) = \frac{1}{H \cdot W \cdot C} \sum_{h,w,c} (\max(E(X_{h,w,c}) - R_{\text{high}}, 0))^2 \quad (5.15)$$

and the lower resistance regularisation term, $r_{\text{lower}}(\cdot)$, is given as:

$$r_{\text{lower}}(X) = \frac{1}{H \cdot W \cdot C} \sum_{h,w,c} (\max(b_{\text{low}} - E(X_{h,w,c}), 0))^2 \quad (5.16)$$

5.3.2.2 Energy Regularisation

The second regularisation term is based on the energy cost function in Equation 5.12 and forms the basis of the trade-off between the use of stable resistance states which are less prone to resistive drift (and information decay) for a given delay, and the image reconstruction error. We specify an energy budget e_b which is the upper limit on the average energy (the power) permitted for use in encoding information. The energy regularisation function $\mathcal{E}(\cdot)$ was given in Equation 5.12.

Making use of $\mathcal{E}(\cdot)$, given its necessary parameters, our second regularisation term, the energy regularisation term r_{energy} , is given as:

$$r_{\text{energy}}(X) = \frac{1}{H \cdot W \cdot C} \sum_{h,w,c} \left(1 - \mathcal{E} \left(\frac{E(X_{h,w,c})}{e_b} \right) \right)^2 \quad (5.17)$$

where X is the image to be encoded, E is the encoder mapping, and $\mathcal{E}_d$ is our delay-parameterised energy regularisation function. Note that we normalise the output energies by the budget, e_b , to achieve a consistent scaling, regardless of the absolute magnitude of the energy budget, in the energy regularisation.

Thus we obtain our loss function to be optimised, shown in Equation 5.18, with parameters λ_{res} and λ_{energy} being design choices used to adjust the stringency of regularisation relative to the main reconstruction error. In our experiments, we set $\lambda_{\text{resistance}} = \lambda_{\text{energy}} = 1$.

$$\begin{aligned} l(X) = & \sqrt{\sum_{i=1}^b (X_i - D(C(E(X_i))))^2} \\ & + \lambda_{\text{resistance}} \left(\sum_{i=1}^b r_{\text{upper}}(X) + \sum_{i=1}^b r_{\text{lower}}(X) \right) \\ & + \lambda_{\text{energy}} \left(\sum_{i=1}^b r_{\text{energy}}(X') \right) \end{aligned} \quad (5.18)$$

5.3.3 Conditional Generalised Divisive Normalisation

GDN is a form of normalisation designed to mimic a canonical computational primitive in the human brain [Burg et al. \(2021\)](#), namely the phenomenon of Divisive Normalisation. Divisive normalisation has been adapted to create **GDN**, for use in neural networks. A form of **GDN** has been used in image compression to improve the learning of an informative representation for source coding of images [Ballé et al. \(2017\)](#). It can be used as an alternative to other kinds of normalisation, such as batch normalisation and layer normalisation. **GDN** has been used in **Deep JSCC** autoencoder architectures [Bourtsoulatze et al. \(2019\)](#) for the transmission of natural images. For a given convolutional output layer index k , the **GDN** transform for a weight $w_i(m, n)$, denoted by u_i - where (m, n) are the spatial dimensions and i indexes the channel dimension - is given as follows [Ballé et al. \(2017\)](#):

$$u_i(m, n) = \frac{w_i(m, n)}{\sqrt{\beta_{k,i} + \sum_j \lambda_{k,ij} (w_j^k(m, n))^2}} \quad (5.19)$$

where the set of parameters $\beta_{k,i}$ and $\gamma_{k,ij}$ are to be learnt. Note that the normalisation uses the square of each weight $w_j^k(m, n)$, ensuring that the magnitude, rather than the sign, is used to normalise the neighbouring weights across the channel dimension.

We can consider delay information to be a separate modality to the information in the image to be compressed. We take inspiration from a conditional adaptation of Batch Normalisation (**BN**), which is known as Conditional Batch Normalisation (**CBN**) [de Vries et al. \(2017\)](#). This technique was originally introduced to allow for the modulation of the processing of visual information (images) using linguistic information (language) to allow for the learning of better representations for Visual Question Answering (**VQA**) problems. We adapt this approach to introduce a conditioning to **GDN**. We term this conditional version **CGDN**.

The **CGDN** transform is shown in Figure 5.4. We use a fully connected processor with nonlinear activations to form an embedding of the delay. Simultaneously, using global average pooling to reduce the dimensionality, along with a fully connected processor, we form an embedding for the activations of the output of the convolutional layer. Both representations are fed to a fully connected combined beta processor network to produce a vector of dimension $\mathbf{R}^{c_k}$, where c_k is the number of channels in layer k of the convolutional architecture. The output vector is then added to the β parameters for that layer, which are indexed by i in Equation 5.19.

5.3.4 Additional Residual Delay Conditioning

To condition the encoder and/or decoder on the delay condition, we also include an additional fully connected residual delay processor, that takes d along with either $\mathbf{R}$ or $\hat{\mathbf{R}}$ (i.e. the latent representation at the output of the encoder, or the input to the decoder) and transforms them into a residual vector of dimension $\mathbf{R}^{c_l \cdot h_l \cdot w_l}$ which is added pointwise to the given resistance input.

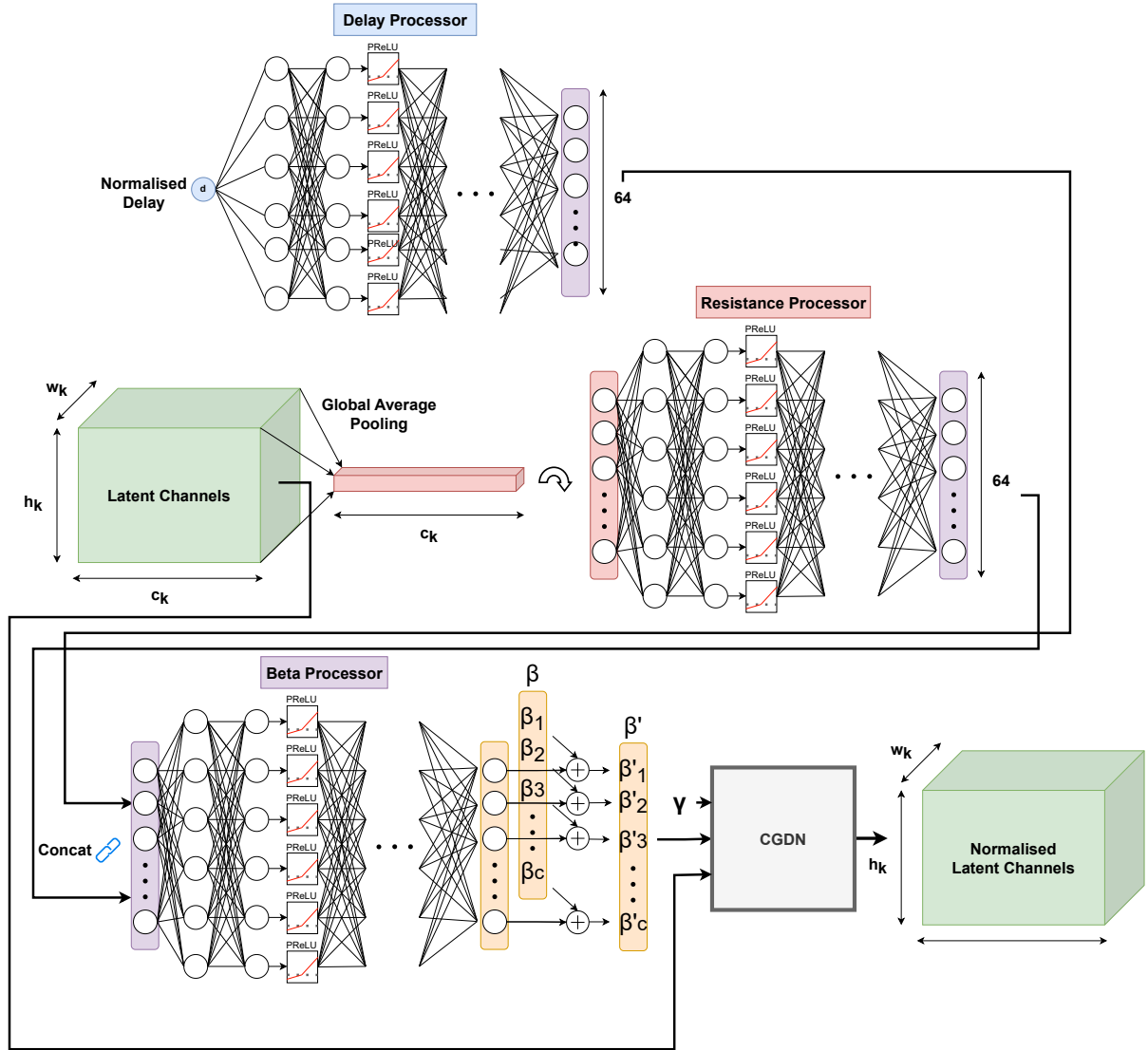


Figure 5.4: The **CGDN** transform for conditioning the encoder and/or decoder network on the delay, applied to the latent channels outputs for each convolutional layer for the encoder/decoder conditioned settings.

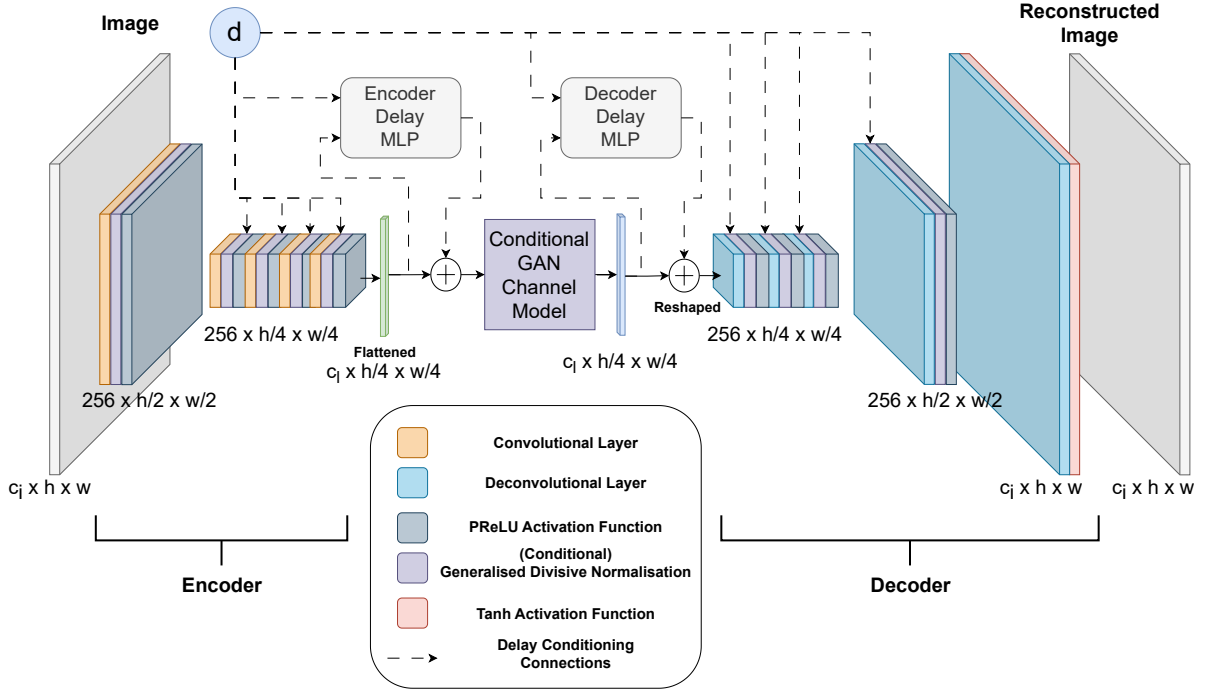


Figure 5.5: The **Deep JSCC** autoencoder architecture used for training, based on modifying an existing architecture to the delay-conditioned setting [Bourtsoulatzte et al. \(2019\)](#). In our model, we replace the **GDN** layers with **CGDN** layers, as well as introducing fully connected residual delay processors at the output of the encoder and/or the input of the decoder, depending on the delay conditioning setting. **CGDN** and the delay processors are only included for the encoder/decoder, respectively, if the setting includes such delay conditioning. For unconditioned settings, we make use of **GDN** transforms and do not include the fully connected delay processors. In the diagram c_i are the number of colour channels in the image, and c_l are the number of latent channels. Changing c_l modifies the compression rate of the autoencoder.

5.3.5 Deep Joint Source-Channel Coder Architecture

The architecture of the autoencoder used for addressing the information storage problem is shown in Figure 5.5. This architecture is based on an existing deep convolutional **Deep JSCC** architecture [Bourtsoulatzte et al. \(2019\)](#), with the **GDN** modules replaced with **CGDN** module, which are passed the delay in order to condition the encoder and/or decoder transform(s) on the delay condition in the case of delay conditioning, as well as the inclusion of residual delay processor fully connected layers at the output of the encoder/input to the decoder, if the encoder/decoder are delay conditioned.

5.3.6 Delay Specialisation

We find that training on batches with a *single delay* condition at a time improves the ability of the autoencoder network to tailor its function to the delay condition. As such, we enforce the energy constraint using a separate set of (fixed) delays, belonging to the set $\mathcal{D}_{\text{energy}}$, which is

defined for a particular batch size b , with delays for a uniform delay distribution in the range $[d_{\min}, d_{\max}]$:

$$\alpha \triangleq \frac{d_{\max} - d_{\min}}{b - 1}$$

$$\mathcal{D}_{\text{energy}} = \{d_{\min}, d_{\min} + \alpha, \dots, d_{\max}\} \quad (5.20)$$

Given each batch of the encoder, we compute the energy regularisation term separately to the other losses, according to the delays $d \in \mathcal{D}_{\text{energy}}$. This ensures that the energy constraint need not be perfectly satisfied for any given random batch of delays sampled to compute the reconstruction error, but must only be obeyed on average according to the uniform delay distribution, which allows the autoencoder to learn to trade off the energy budget among different delay conditions

5.3.7 Training Parameters

We saw in Section 4.4.3 that the **cGAN** could reasonably match the statistics not only within the bounds of the delays present in the training data, but also up to the bounds of the delays supplied considered during delay discrimination (i.e. $[1, 500]$). In order to train the autoencoder for delays larger than this maximum value, we choose to evaluate the **cGAN** recurrently n times, extending the validity of the delay conditions for the model to $500n$. We trade off the value of n for the computational complexity and risk of vanishing gradients when evaluating the **cGAN** recurrently, choosing a relatively conservative value of $n = 4$, ensuring that the model is valid for delays up to 2000, which is within the range of delays we choose to train for: $d \in [0, 1000]$.

We set the batch size $b = 32$ for training. We train using an Adam optimiser **Kingma and Ba (2017)**, with a learning rate of 5×10^{-5} . We train each delay conditioned scenario (see below) for 50 epochs on the CIFAR-10 dataset **Krizhevsky (2009)**, optimising the **Deep JSCC** autoencoder to minimise the **MSE** of reconstruction of the images.

5.4 RESULTS

We train the autoencoder using four different experimental settings. The task is to reconstruct the data, across the given conditioning delay distribution. The network learns to minimise the error for the given delay distribution, learning the best distribution of resistances to program that will result in a minimum expected distortion.

1. In the “unconditioned” setting, we train the encoder and decoder without knowledge of the delay. Thus, the autoencoder must learn the best “average” encoding and decoding scheme that minimises the error over the given range of delay conditions.
2. In the “encoder conditioned” setting, the encoder is passed information about the delay, through the use of the aforementioned **CGDN** modules and the encoder fully connected

residual network. In this setting, the encoder must apply a pre-transform to the input distribution such that using a single decoding strategy, the [MSE](#) is minimised.

3. In the “decoder conditioned” setting, the decoder is passed information about the delay using [CGDN](#) modules and the decoder fully connected residual network, while the encoder remains delay agnostic. The decoder must work to correct for distortion introduced, given its knowledge about the delay. The encoder must learn an optimal input distribution to enable the decoder to perform optimally.
4. Finally, in the “both conditioned” (or simply “conditioned”) setting, both the encoder and decoder networks are passed delay information using [CGDN](#) modules and their respective conditional residual networks. In this setting, a delay-dependent encoding and decoding strategy can be learnt for each individual delay.

Note that the “encoder conditioned” and “decoder conditioned” settings are supersets of the “unconditioned” setting. The “both conditioned” setting is in fact a superset of every other setting, and thus theoretically has the capacity to perform at least as well as all other settings, if the delay information can be exploited correctly by the network during training.

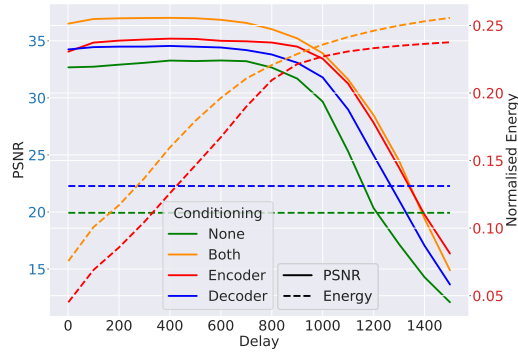
5.4.1 *Reconstruction Results*

Figure [5.6](#), shows the reconstruction errors for a variety of delay conditions, for each of the conditioning settings. We can see from the figure that for energy budgets $e_b \geq 0.5J$, the networks are able to learn effectively using less than the required energy budget, however the delay conditioning of the networks still benefits training. The disparity in the energy budgets stems from the encoder making use of different encodings, dependent on the delay, however energy is not a bottleneck.

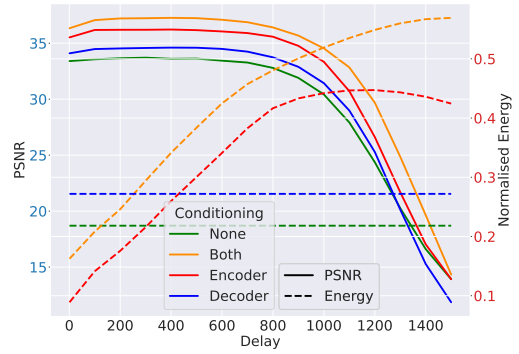
We see that as the budget falls to $0.1J$, energy is maximised, making it clear that the energy budget begins to become a limiting factor for the performance. For $0.05J$ and $0.01J$, we see steady decreases in the average Peak Signal to Noise Ratio ([PSNR](#)), which is defined as $10 \log_{10}(1/MSE)$, with the delay-conditioned networks consistently outperforming the unconditioned network. In the case of decoder conditioning, we see that the network is able to better adjust for the deterministic distortions introduced by the channel, improving performance, however we observe that the introduction of encoder conditioning is the most beneficial, in terms of the reconstruction error, while conditioning of both the encoder and the decoder produces a synergistic effect that results in a higher performance than is possible with each of the individual networks, in most cases.

We see that the encoder network is able to generalise to delays outside of the distribution given. We see that the energy constraints are violated in order to ensure that enough information is retained, following storage and recovery, to reconstruct the image to a reasonable degree. The violation of the energy constraints demonstrates that the trade-off that the network has learnt between energy consumption and storage reliability generalises to larger delays.

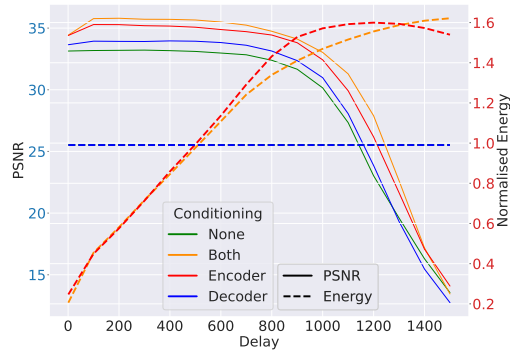
Figure [5.7](#) shows image reconstruction results under a range of channel delays, for the different delay-conditioned settings studied, for an energy budget of $0.1J$. Figure [5.8](#) shows



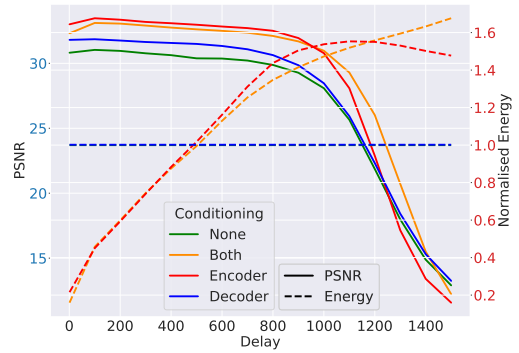
(a) Energy budget 1.0J.



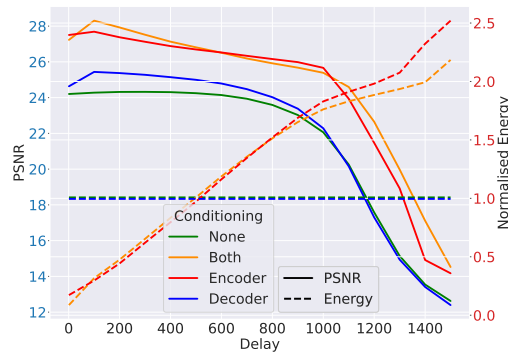
(b) Energy budget 0.5J.



(c) Energy budget 0.1J.



(d) Energy budget 0.05J.



(e) Energy budget 0.01J

Figure 5.6: A comparison of the reconstruction errors for a variety of autoencoder experiments, demonstrating the impact of delay conditioning of the encoder/decoder networks. Introducing delay conditioning to the encoder, allows the autoencoder to allocate different energy budgets to different conditions, while still respecting the average energy budget. Decoder conditioning also results in improved results compared to the absence of conditioning, with the decoder able to correct for delay-dependent distortions, resulting in better SNR reconstruction values across a wider range of the delays considered.

similar results for a reduced budget of $0.01J$. We see that reconstruction quality generally improves with decreasing delay, except in the case of the unconditioned network, which uses a single encoding and decoding scheme, tailored to the noisiest setting. Therefore, particularly for very low energy budgets, conditioning is essential for achieving a reasonable reconstruction quality.

5.4.2 *Single Delay and Noiseless Ablation Experiments*

We compare the performance of the delay conditioned experiments for an energy budget of $0.01J$ to the performance of autoencoders trained for specific delay conditions, as well as those trained with a noiseless channel. The results of this experiment are shown in Figure 5.9.

Interestingly, we see that the performance of the unconditioned network is equivalent to that of the network trained only with a delay of 1000, and this is the setting which achieves the best performance on average across the range of delays considered. This suggests that in the case of a budget of $0.01J$, the encoding and decoding strategy is to code for the worst case noise. It should also be noted that the performance of the networks trained specifically for the delays 1, 10, and 100, exceeds the performance of the encoder and decoder conditioning for those particular delays. However, it is only for these delays that the network performance is improved, and is due to the specialisation to these particular channel conditions - a much less complex problem than the conditioned scenarios. We see that in the case of training with no channel noise whatsoever, the performance is similar to the performance when a delay of 1 is used for training. We see that in the cases of the single-delay conditioned experiments, the performance across a wider range of delays is improved for training on larger delays, which is due to the need for the network to use a more general encoding and decoding strategy that has increased resilience to the higher level of distortion associated with larger delays.

5.4.3 *Energy Distribution*

Figure 5.10 shows a comparison of the encoded distributions of (unnormalised) resistances for a variety of energy budgets and delay conditions in the encoder conditioned experiments. We see that using delay conditioning, the encoder is able to tailor the input distribution to the delay, in order to better distribute the available energy budget, and to improve performance for larger delays by increasing the power allocation, at the expense of reduced power allocation for lower delay settings, where there is less noise. Note that many histograms show peaks at the equilibrium resistance value. In such cases, we can hypothesise that the network chooses to use a sparse encoding strategy, where a few of the memristors are allocated a majority of the information content, along with a much higher energy budget.

Note that the delay conditioning allows the autoencoder to not only account for the different deterministic transformations introduced by different delays, but also to adopt different encoding schemes at different PSNR levels. For example, it may choose to expend more energy on encoding information at a variety of lower, relatively stable resistance levels for small delays, under which conditions the states will be relatively stable. In contrast, in the high-delay setting,

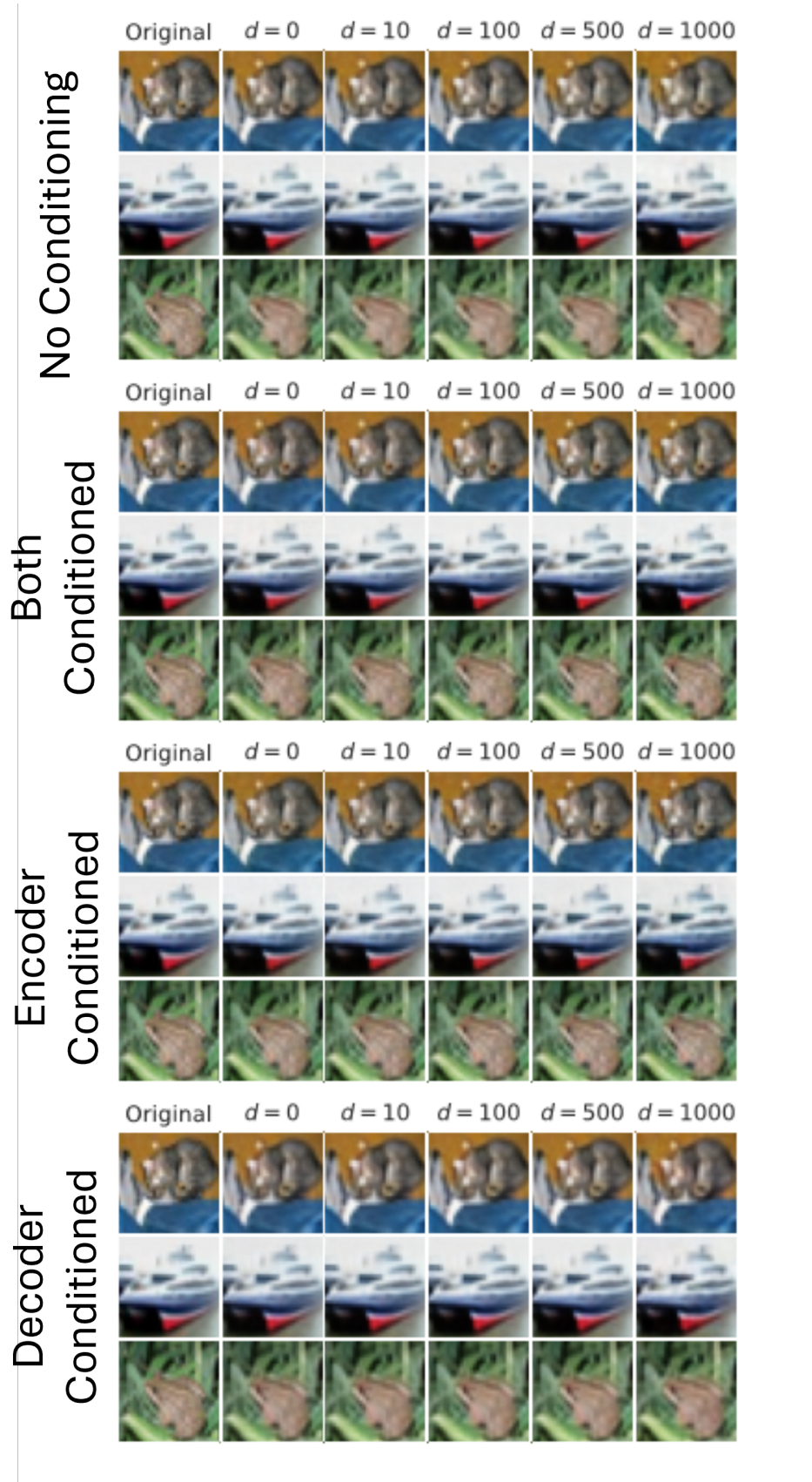


Figure 5.7: Image reconstruction results for a range of delays, for the different delay conditioned settings, for an energy budget of 0.1J.

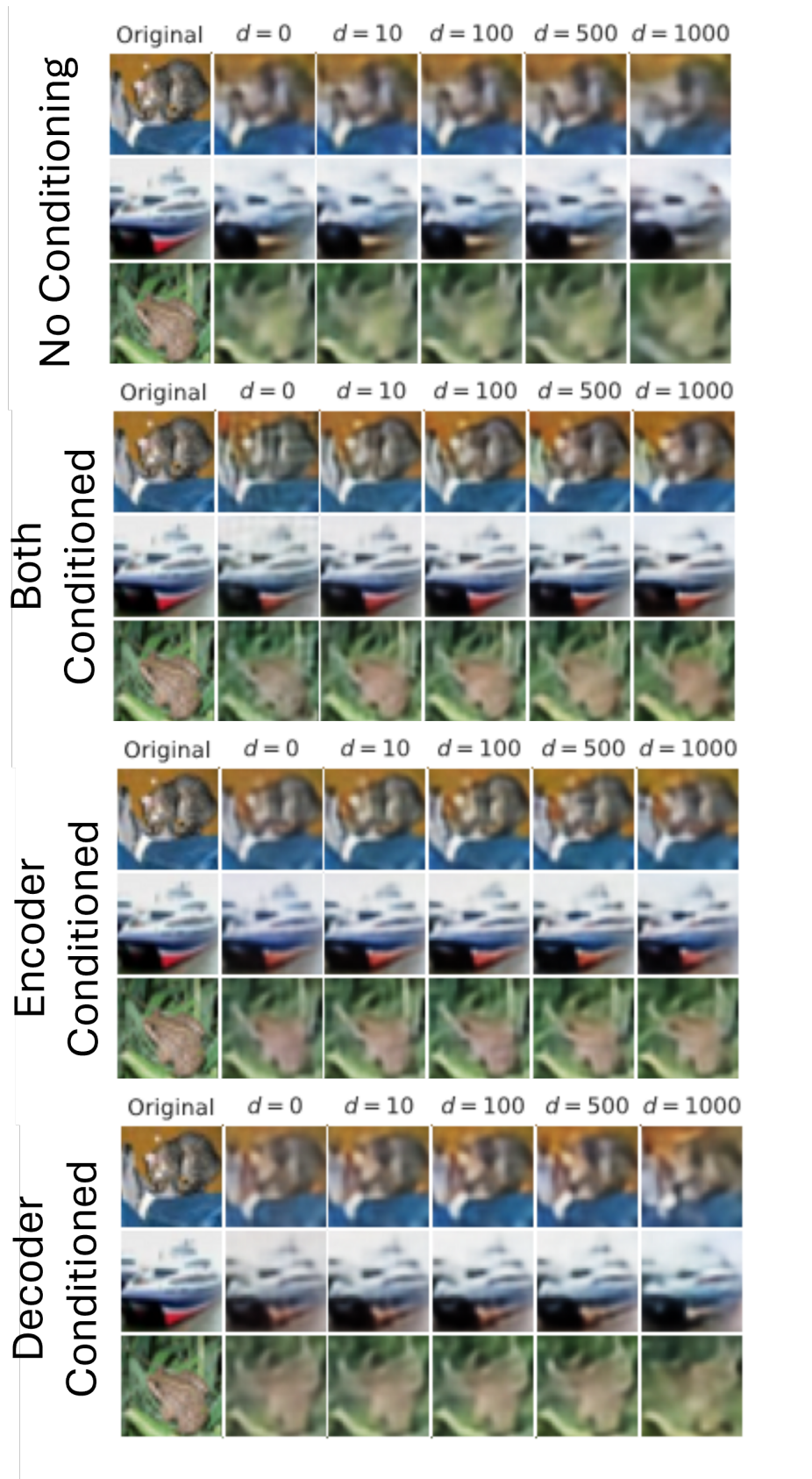


Figure 5.8: Image reconstruction results for a range of delays, for the different delay conditioned settings, for an energy budget of 0.01J.

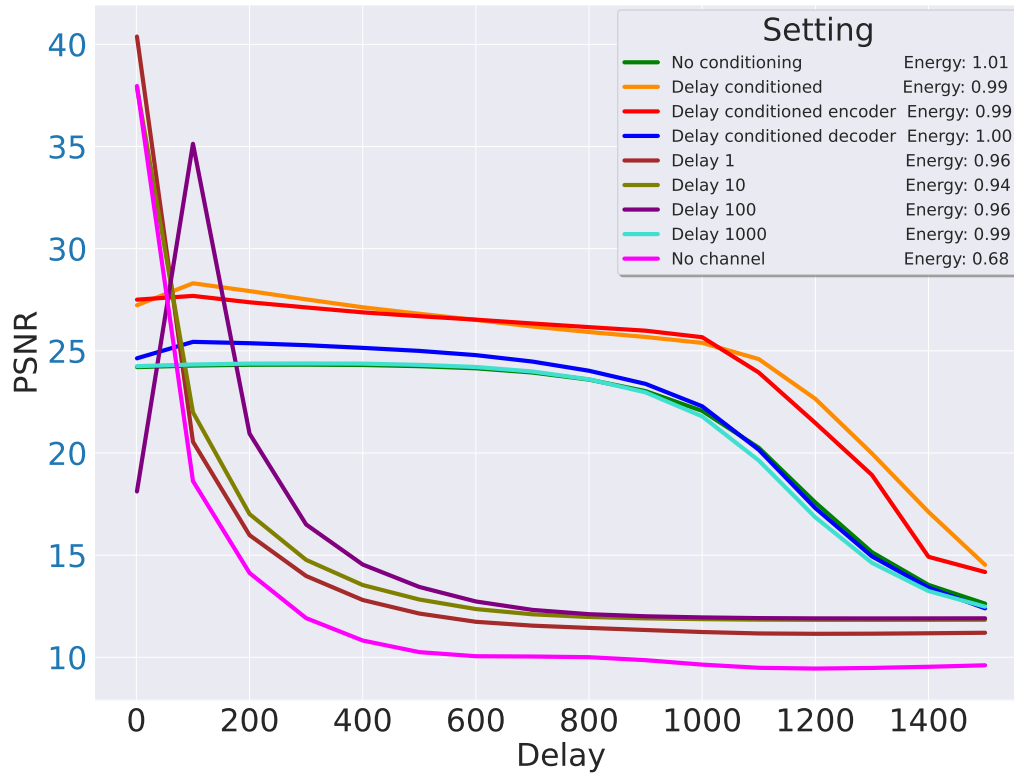
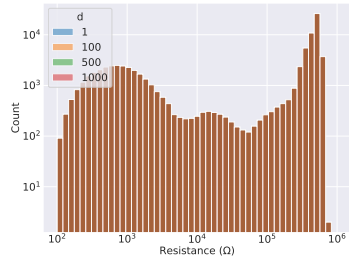
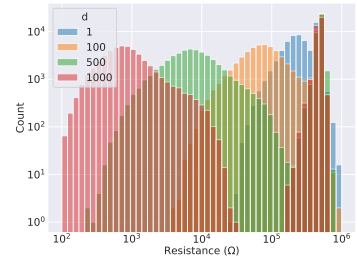


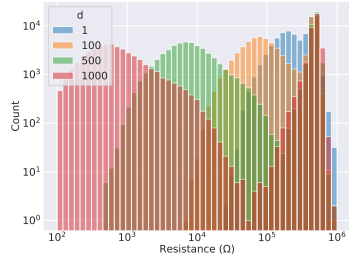
Figure 5.9: Comparison of the encoder/decoder/encoder and decoder -conditioned settings for autoencoder training to a range of ablation experiments, for training, without delay conditioning, on single delays only (1, 10, 100, 1000) and on a noiseless channel. We see that although training for a particular channel setting (delay) can improve the performance for that particular condition, the performance is much worse across other delays. The legend displays the average energy consumption for each of the scenarios.



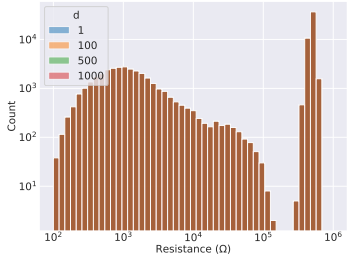
(a) No delay conditioning. Energy budget 0.1J.



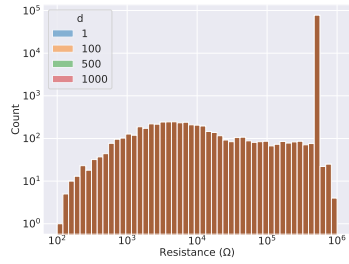
(b) Delay conditioned encoder/decoder. Energy budget 0.1J.



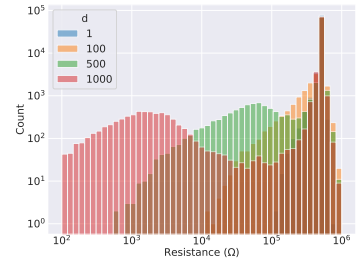
(c) Delay conditioned encoder. Energy budget 0.1J.



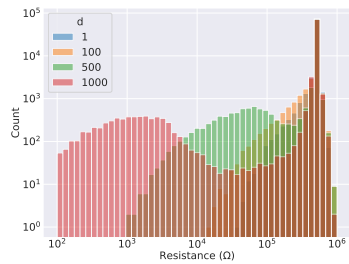
(d) Delay conditioned decoder. Energy budget 0.1J.



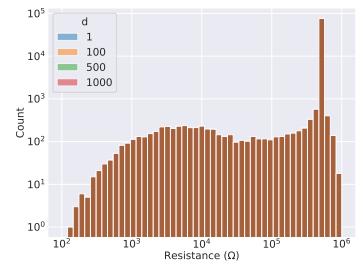
(e) No delay conditioning. Energy budget 0.01J.



(f) Delay conditioned encoder/decoder. Energy budget 0.01J.



(g) Delay conditioned encoder. Energy budget 0.01J.



(h) Delay conditioned decoder. Energy budget 0.01J.

Figure 5.10: A comparison of the input distributions for the different encoder/decoder conditioned settings, for two different energy budgets: 0.1J, and 0.01J. We see that in the unconditioned cases, and the decoder conditioned cases, the histograms are the same across delays, whereas encoder and decoder conditioning leads to adaptation of the input distribution, with input distributions that are more costly in terms of the energy budget being allocated to larger, noisier, delay conditions. Note that the counts are shown on a logarithmic scale for improved clarity.

it may choose to instead expend energy on maximising the distance between a smaller number of discrete input symbols that will be more robust for the lossier channel output.

5.4.4 Ground Truth Model Evaluation

In order to evaluate the efficacy of the **cGAN** model for simulating the storage of information in the presence of resistive drift, we evaluate the autoencoder on the ground truth, event-based model, as a measure of its ability to generalise to the statistics of a real device, following training using a generative model. This also serves as a measure of the efficacy of the delay-conditioned **cGAN** for simulating the true statistics of the ground truth model (in this case, the event-based model developed in Chapter 3).

Figure 5.11 shows the performance of the energy constrained autoencoders when evaluated on the ground truth event-based model, for the different delay-conditioning settings, for an energy budget of 0.01J and for delays in the range [1, 1000], alongside an equivalent evaluation on the **cGAN**. We see that the performance when evaluating on the ground truth model is comparable to that when evaluating on the **cGAN** model for most delays, demonstrating that the **cGAN** is able to accurately model the conditional drift distribution, for a wide range of delay conditions, in order to enable the autoencoder to learn to store information on the devices in the presence of drift noise.

5.5 CONCLUSION AND FUTURE WORK

In this chapter, we have shown that a joint source-channel coder can be trained on a differentiable model of resistive drift, such as that developed in Chapter 4, in order to learn to jointly source and channel code for data storage on an array of memristive devices, with a graceful trade-off between the number of devices used and the reconstruction error in terms of a chosen metric (such as the **MSE**). We demonstrated that conditioning on the delay could be achieved in order to enable the encoder and decoder to adapt the energy consumption and channel input distribution to different delay settings for improved performance, while still satisfying average power constraints.

It should be noted that though the proposed energy trade-off does broadly characterise the relationship between the energy expended for storage and magnitude of the resultant states, more specific models for real programming settings may need to take into account additional factors such as the uncertainty in the final state achieved (due to programming noise) as well as the potential need for multiple iterative passes to program a state to a given degree of accuracy. This can be explored as future work. Future work could also include using the conditioning approaches developed to code for other forms of noise. **CGDN** could be investigated as a technique used to condition on other channel characteristics, such as the **SNR** of a channel, in more generic communication settings.

In the next chapter, we will move on from the problem of noisy storage, to one of noisy computation, looking at one possible way in which adversarial noise can be used to benefit information processing, in the context of a generative modelling problem.

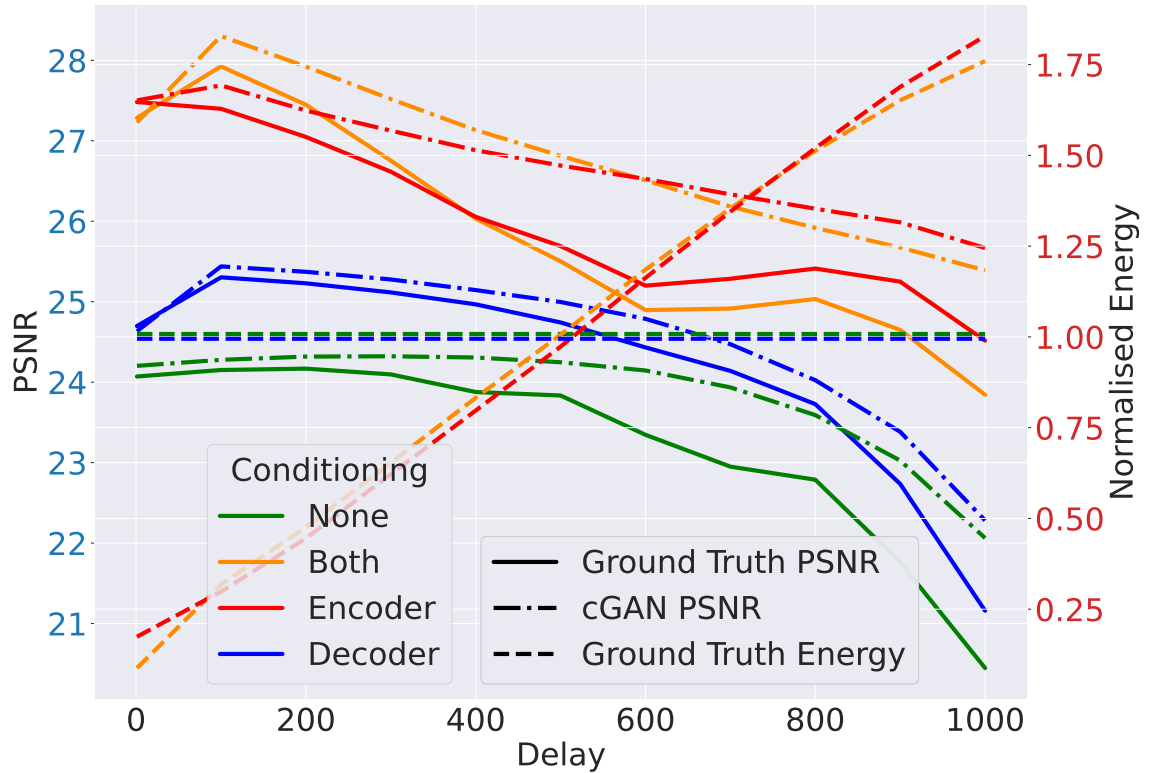


Figure 5.11: Evaluation on the ground truth event-based model for the reconstruction PSNR of the autoencoders trained for an energy budget of 0.01J on the cGAN model from Chapter 4. We also show the energy consumption of the input distribution for each delay and input resistance condition. We plot the performance when evaluated on the cGAN model alongside these results for comparison. We see that for some delays, there is drop in performance when we switch from evaluation on the trained cGAN model to evaluation on the ground truth event-based model, highlighting regions where the statistics were not matched perfectly. However, the overall match in the level of performance is seen to be good.

ADVERSARIAL JAMMING FOR AUTOENCODER DISTRIBUTION MATCHING

The concept of competition is both intriguing, and also prevalent throughout the world of intelligent beings. Competition has been shown to be a mechanism by which systems can optimise certain behaviours in a variety of settings. The most fundamental, perhaps, is the process of evolution, where biological organisms compete for limited resources, by the process of natural selection, leading to adaptation to improve survival rates. The same principle exists in the economic system of capitalism, where multiple parties compete to offer the most desirable product at the best price to consumers, resulting in optimisation of the usefulness and/or cost of the product in a distributed way. Neurons in the brain also compete through the process of Hebbian competition, which can motivate the specialisation of neurons to respond to different inputs [White \(1992\)](#), through mechanisms such as lateral inhibition [Purves \(2018\)](#).

In this chapter, building upon the idea that noise and adversarial noise can be used to improve neural network training and robustness, that was introduced in Section 2.6 of Chapter 2, we take inspiration from the idea of competition as a fundamental computational primitive and aim to explore how computation in the presence of adversarial noise can be exploited as a computational feature, rather than simply an inconvenience. We, again, approach this from the perspective of communication theory and propose the use of a communication-theoretic minimax game of adversarial wireless jamming to regularise the latent space of an autoencoder to match a diagonal Gaussian distribution. The game involves the minimisation of a Mean Squared Error ([MSE](#)) distortion, where a jammer attempts to disrupt the transmission and subsequent recovery of a Gaussian source that is transmitted over an adversarial channel. A consequence of existing theoretical results is shown to be the fact that the saddle point of a minimax game - involving such an encoder, its corresponding decoder, and an adversarial jammer - consists of diagonal Gaussian noise output by the jammer. We use this result as inspiration for a novel approach to distribution matching in the latent space, utilising jamming as an auxiliary objective to encourage the aggregated latent posterior of an autoencoder to match a diagonal Gaussian distribution. Using this new technique, we achieve distribution matching comparable to standard Variational Autoencoders ([VAEs](#)) and to Wasserstein Autoencoders ([WAEs](#)). The proposed approach could also be generalised to enforce the matching of other distributions.

6.1 INTRODUCTION

Generative modelling can be described as the task of drawing artificial samples from an underlying distribution based on the observation of a real set of samples from that distribution. The distribution can be simple and tractable, or it can be complex and intractable. A variety of methods for generative modelling in the field of machine learning and deep learning

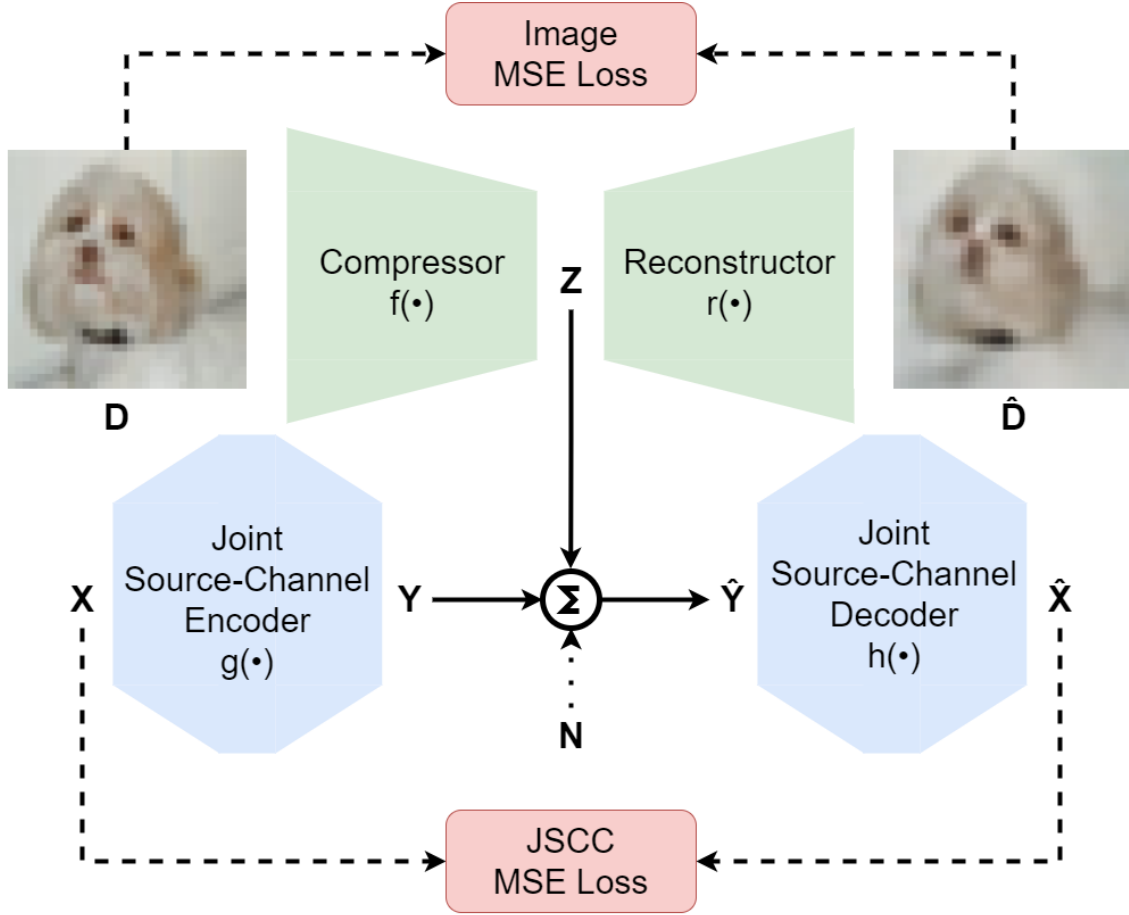


Figure 6.1: Adversarial Jamming (AJ) for prior distribution matching. The game between a compressor/jammer f (compressing the images) and a Joint Source-Channel Coding (JSCC) autoencoder (g, h) (for source $\mathbf{X} \sim \mathcal{N}(\mathbf{o}, \mathbf{I})$) has a saddle point with $\mathbf{Z}^* \sim \mathcal{N}(\mathbf{o}, \mathbf{I})$, implicitly imposing this prior on the latent space.

have been proposed. These include diffusion models [Ho et al. \(2020\)](#), Generative Adversarial Networks (GANs) [Goodfellow et al. \(2014\)](#), VAEs [Kingma and Welling \(2014\)](#), normalising flows [Dinh et al. \(2016\)](#), and energy-based models [Bond-Taylor et al. \(2022\)](#), as mentioned in Section 2.5 of Chapter 2. Many of these can generate samples from the modelled distribution given a representation (invert the mapping). However, the ability to map a sample (either real or generated) back to an appropriate representation is unique to symmetric techniques, namely VAEs and normalising flows.

In this chapter, we focus on VAEs [Kingma and Welling \(2014\)](#), which use a variational approximation to model an intractable prior distribution for a particular data distribution. Commonly, a parameterised multivariate Gaussian distribution is used as the prior. This is because the Kullback-Leibler Divergence (KL Divergence) - in the case of a multivariate Gaussian - has an analytic differentiable form, which can be used as part of the loss function. Secondly, using a known distribution such as a multivariate Gaussian can allow us to sample from the data distribution by first sampling this prior and then generating samples from the resultant posterior using the autoencoder's decoder.

Adversarial learning in the context of neural networks became popular following the seminal paper introducing GANs Goodfellow et al. (2014). Later works have explored adversarial attacks and studied the robustness of networks trained under adversarial noise in a variety of contexts Jiang et al. (2020); Kim et al. (2020); Sun, Zhu, Zhang, Xiong and Zhou (2021). Adversarial learning has been applied in the context of deep learning for communications, for example in the task of neural channel coding for the purposes of improving the performance of a decoder Nguyen et al. (2021) and in generative channel modelling O'Shea et al. (2018); Yang et al. (2019); Ye et al. (2020).

In this chapter, we propose a new AJ approach to distribution matching based on a minimax game between a jammer, $f(\cdot)$, and an auxiliary joint source-channel coder, $g(\cdot)$ and $h(\cdot)$, respectively, for a diagonal Gaussian source, $\mathbf{X}$, with an MSE distortion measure for reconstruction under constraints on the power of both the jammer and the JSCC encoder/decoder (see Figure 6.1). Existing theoretical results concerning the optimal (worst-case) additive jamming noise for such a setting Akyol and Rose (2013); Akyol et al. (2013); Basar and Wu D. (1985) imply that under these conditions, the optimal jammer output will be a diagonal Gaussian random vector to maximise the expected value of the reconstruction error. We exploit this fact in order to produce an AJ regularisation term for an autoencoder that tries to simultaneously minimise the distortion in compressing and reconstructing images from a dataset, while also matching a diagonal Gaussian prior. We show that this technique is effective for distribution matching in the latent space, comparable to a VAE and a WAE. To the best of our knowledge, this is a novel result that exploits a communication theoretic saddle point result to regularise the latent space of an autoencoder for distribution matching, which can lead to similar, potential future extensions.

The context of our generative modelling approach as an alternative regularisation term for distribution matching of the latent space of generative autoencoders is illustrated in Figure 6.2.

Notation. We use $\mathbf{0}$ and $\mathbf{I}$ to represent the k -dimensional zero vector and $(k \times k)$ -dimensional identity matrix, respectively. Also, $\mathbf{R}_A = \mathbf{Q}_A \mathbf{\Lambda}_A \mathbf{Q}_A^T$ denotes the eigenvalue decomposition of the covariance matrix $\mathbf{R}_A$ of random vector $\mathbf{A}$ and $F_A(\omega)$ denotes its characteristic function.

6.2 PROBLEM DEFINITION

Let $\mathbf{X} \in \mathbf{R}^k$, $\mathbf{D} \in \mathbf{R}^n$, and $\mathbf{N} \in \mathbf{R}^k$ be random vectors representing the source distribution, jammer's source of randomness, and channel noise, respectively. Let $f : \mathbf{R}^n \rightarrow \mathbf{R}^k$, $g : \mathbf{R}^k \rightarrow \mathbf{R}^k$, and $h : \mathbf{R}^k \rightarrow \mathbf{R}^k$ be Borel-measurable functions. These respectively represent the jammer, with output $\mathbf{Z} := f(\mathbf{D})$; the encoder (transmitter), with output $\mathbf{Y} := g(\mathbf{X})$; and the decoder (receiver), with output $\hat{\mathbf{X}} := h(\hat{\mathbf{Y}})$, where $\hat{\mathbf{Y}}$ defines the additive noise channel mapping $\hat{\mathbf{Y}} = \mathbf{Y} + \mathbf{N} + \mathbf{Z}$. Then, the zero-sum, minimax game among the transmitter, receiver, and adversarial jammer is shown below with cost function $J(f, g, h)$, and power constraints P_t and P_a for the transmitter and jammer, respectively.

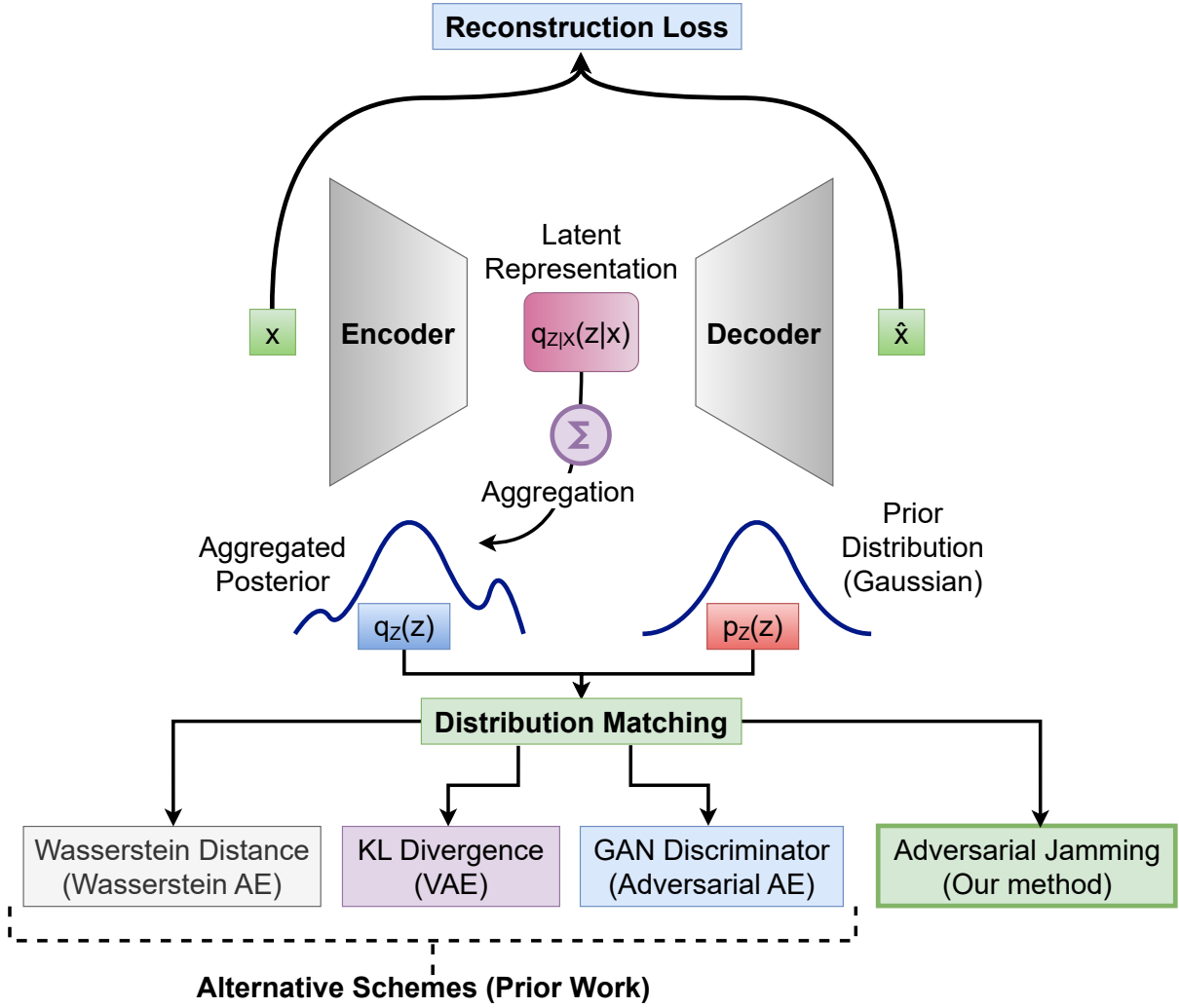


Figure 6.2: AJ can be seen as an alternative regularisation term in the framework of distribution matching regularisation for generative autoencoders. We derive the worst case additive jamming noise for an adversarial game between a transmitter/receiver pair and an adversarial jammer, based on theoretical properties of the noise that are given in Akyol and Rose (2013). Using the learned representations as jammer noise, we enforce a game whose equilibrium is such that the representations match the required Gaussian prior distribution.

$$\max_f \min_{h,g} J(f, g, h) \quad (6.1)$$

$$\text{where } J(f, g, h) = \mathbb{E}[(\mathbf{X} - \hat{\mathbf{X}})^2],$$

$$\mathbb{E}[\|g(\mathbf{X})\|^2] \leq P_t, \text{ and } \mathbb{E}[\|f(\mathbf{D})\|^2] \leq P_a$$

The optimal combined policy for the transmitter, receiver, and jammer is the saddle point solution to the zero-sum game, (f^*, h^*, g^*) , where the two inequalities in Equation 6.2 are simultaneously satisfied.

$$J(f, h^*, g^*) \leq J(f^*, h^*, g^*) \leq J(f^*, h, g) \quad (6.2)$$

It was shown in [Akyol et al. \(2013\)](#) that the problem of optimal (zero-delay) jamming over a scalar additive noise channel (i.e. $k = 1$) is closely connected to the linearity of g^* and h^* . Where possible, a jammer will generate additive noise that forces the optimal transmitter and receiver to be linear transformations (the “matching condition”). In the case of a scalar Gaussian source being sent over a Gaussian channel, the output of the jammer will also be Gaussian.

The scalar jamming result was extended and generalised in [Akyol and Rose \(2013\)](#), where it was shown that a vector version of the matching condition can be used to find saddle points for the vector version of the minimax game. An optimal encoding function for the transmitter will be a linear transformation (followed by a random sign change through multiplication by a Bernoulli random variable with $p = 0.5$ over the alphabet $\{0, 1\}$). It was shown that in the vector setting, the power allocation of the transmitter will obey reverse water-filling.

Our focus in this work is on the nature of the optimal jamming function. The necessary and sufficient condition for the linearity of optimal estimation - and a requirement for the optimal jamming strategy - is given in Equation 6.3, where Σ is a diagonal power allocation matrix. The optimal jammer also obeys water-filling, dependent on the eigenvalues of $\mathbf{R}_N$ [Akyol and Rose \(2013\)](#).

$$\frac{\delta \log F_{\Sigma Q_X^T X}(\omega)}{\delta \omega_i} = S_i \frac{\delta \log F_{Q_Z^T (N+Z)}(\omega)}{\delta \omega_i}, 1 \leq i \leq m \quad (6.3)$$

where $S = \Sigma \Lambda_X \Sigma \Lambda_Z^{-1}$

Such a characterisation of the nature of the optimal jamming solution implies that if $\mathbf{X} \sim \mathcal{N}(\mathbf{o}, \mathbf{I})$ and $\mathbf{N} \sim \mathcal{N}(\mathbf{o}, \mathbf{I})$, then the optimal jammer output will also be an isotropic diagonal Gaussian distribution $\mathbf{Z} \sim \mathcal{N}(\mathbf{o}, \kappa \mathbf{I})$, where κ is a scalar constant, to match the source distribution and enforce the transmitter’s linearity. This can be easily verified by noting that, in this special case, $\mathbf{I} = \alpha \Sigma = b \Lambda_X = c \Lambda_Z = d Q_Z = e Q_X$, where α, b, c, d, e are scalar constants. Thus we have a solution $\mathbf{Z} = \beta(\mathbf{N} - \mathbf{X})$, where β is scalar, which means that a saddle point solution for

$\mathbf{Z}$ is also a diagonal Gaussian random vector. This saddle point solution to the minimax game is also (almost surely) unique [Akyol and Rose \(2013\)](#).

6.2.1 Jamming as Regularisation for an Autoencoder

We propose the use of this communication theoretic game to regularise the latent space of an autoencoder. Consider a data distribution random vector, $\mathbf{D} \in \mathbf{R}^n$; for example, flattened image data with $n = l \cdot w \cdot m$, where l , w , and m are the length, width, and colour channels of the images, respectively. $\mathbf{D}$ shall be the source of randomness for our jammer. We introduce an auxiliary Deep Joint Source-Channel Coding ([Deep JSCC](#)) autoencoder that attempts to transmit and recover a source $\mathbf{X} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ over the additive noise channel. [Deep JSCC](#) has been shown to be an effective technique for transmitting complex source distributions, such as natural images, over unknown communication channels [Bourtsoulatze et al. \(2019\)](#); [Xu et al. \(2021\)](#); [Ye et al. \(2018\)](#) and continues to be a topic that is actively researched in the field of semantic communication [Xu et al. \(2023\)](#).

Since we deal with two autoencoders, one operating on data $\mathbf{D}$ and one operating on Gaussian source $\mathbf{X}$, we use the terms “data autoencoder” and “[Deep JSCC](#) autoencoder” to distinguish between the two. We also refer to the encoder/decoder of each as the compressor/reconstructor and the transmitter/receiver, respectively. In the context of the vector jamming game as defined above, the reconstructor is f , the transmitter is g , and the receiver is h . The reconstructor will be a function $r : \mathbf{R}^k \rightarrow \mathbf{R}^n$ with output $\hat{\mathbf{D}} := r(\mathbf{Z})$.

The objective of the [Deep JSCC](#) autoencoder is to minimise the reconstruction error of the Gaussian source, $\mathcal{L}_{\text{jsc}}$, given in Equation 6.4. In turn, the objective of the data autoencoder is to minimise an objective function, $\mathcal{L}_{\text{data}}$, which is composed of two terms: its own reconstruction error for the data distribution, as well as the negative of $\mathcal{L}_{\text{jsc}}$, as shown in Equation 6.5 below. Here, $\mathcal{L}_{\text{data}}$ mirrors the objective of a [VAE](#), with $-\mathcal{L}_{\text{jsc}}$ replacing the usual [KL Divergence](#) term.

$$\mathcal{L}_{\text{jsc}}(g, h) = \mathbf{E}[(\mathbf{X} - \hat{\mathbf{X}})^2] \quad (6.4)$$

$$\mathcal{L}_{\text{data}}(f, r) = \mathbf{E}[(\mathbf{D} - \hat{\mathbf{D}})^2] - \eta \mathbf{E}[(\mathbf{X} - \hat{\mathbf{X}})^2] \quad (6.5)$$

where η is a positive regularisation constant.

We enforce power constraints $P_t = P_a = 1$ on the outputs $\mathbf{Y}$ and $\mathbf{Z}$ by normalising them (in practice, in mini-batch training with Stochastic Gradient Descent ([SGD](#)), by using the empirical batch statistics). The competition between the compressor, acting as a jammer, and the [Deep JSCC](#) autoencoder, sets up a communication game in the form seen in Equation 6.1. The saddle point of this communication game is for the output of the jammer $\mathbf{Z}$ to be an isotropic diagonal Gaussian. Our proposed approach makes use of the maximisation term from the game in Equation 6.1 as a regularisation term that encourages the latent distribution to match this distribution. The proposed scheme is illustrated in Figure 6.1. For our experiments,

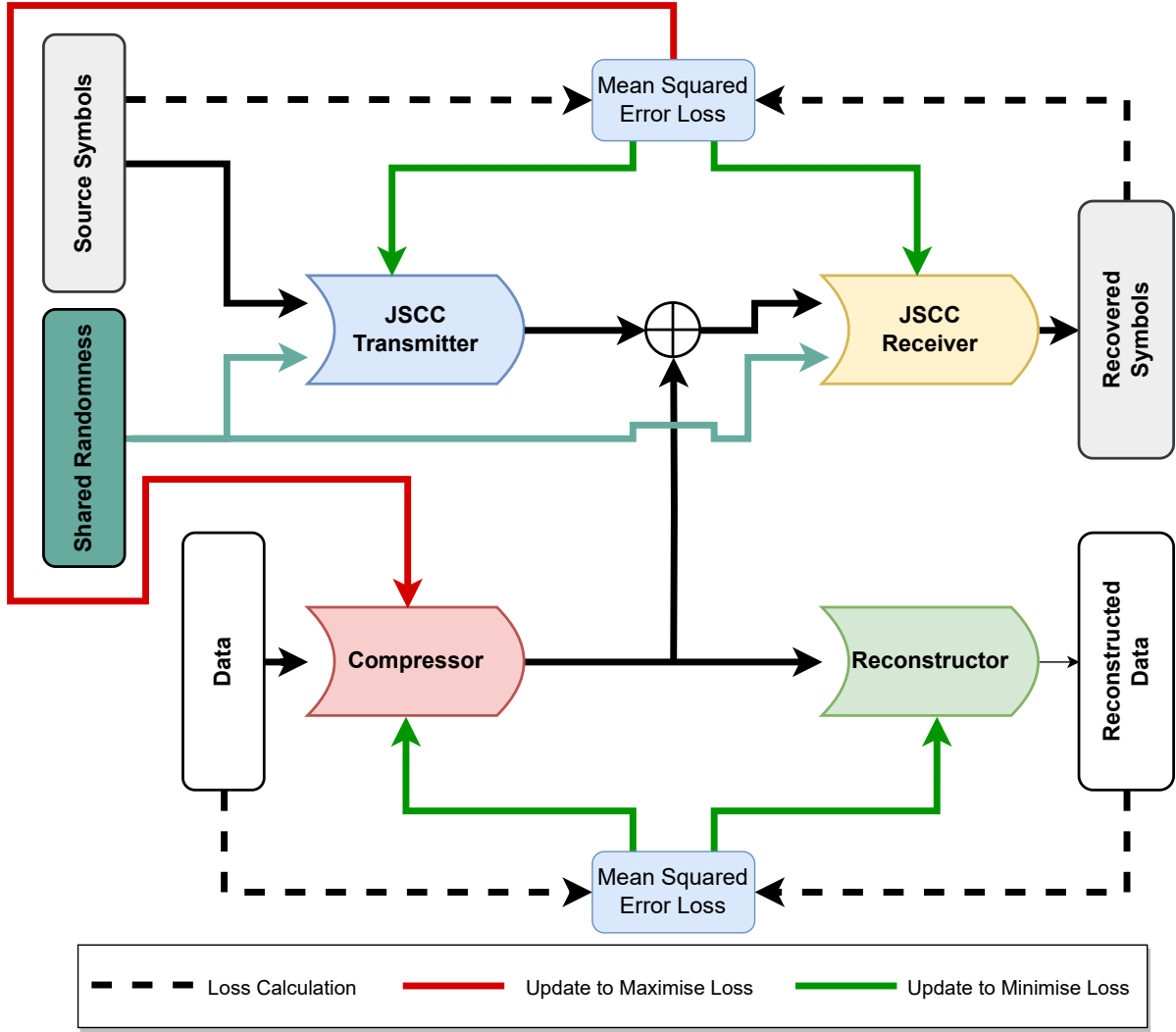


Figure 6.3: Framework used for regularisation using the AJ problem. The specific architecture for the data autoencoder (compressor and reconstructor) is detailed in Figures 6.4. We make use of a residual and symmetric fully connected autoencoder architecture with residual connections for the Deep JSCC autoencoder (transmitter and receiver) as detailed in Section 6.3.3.

we zero out the power of the fixed channel noise, N , using only the adversarial jamming noise, Z , to disrupt communications.

6.3 METHOD

The framework used for training the autoencoder is shown in Figure 6.3, showing the concatenation of shared randomness to the input source vector for the transmitter and receiver.

6.3.1 Data Autoencoder Architecture

We use the same architecture for our compressor and reconstructor networks in all experiments, as shown in Figure 6.4. We vary the number of layers and filters in the architecture, depending

Table 6.1: Architectural parameters for the data autoencoder, for each dataset trained on.

Dataset	c	filter size	n
MNIST	128	5×5	3
CIFAR-10	512	5×5	3
CelebA	1024	5×5	4

on the dataset used (not the algorithm) (see Section 6.3.7), with parameters as given in Table 6.1. Note that for the VAE experiments, in addition to the given convolutional architectures, we use a fully connected readout layer at the compressor output to map to values of $\sigma_{Z|X}^2$ and $\mu_{Z|X}$.

We use an architecture similar to that of Deep Convolutional Generative Adversarial Network (DCGAN) Radford et al. (2016) convolutional autoencoder architecture for our data autoencoder, with output normalisation to ensure adherence to constraints on the variance and mean. k is the length of the representation - the dimension of the adversarial noise $\mathbf{Z}$ output by the compressor. k is varied for each experiment, depending on the dataset used.

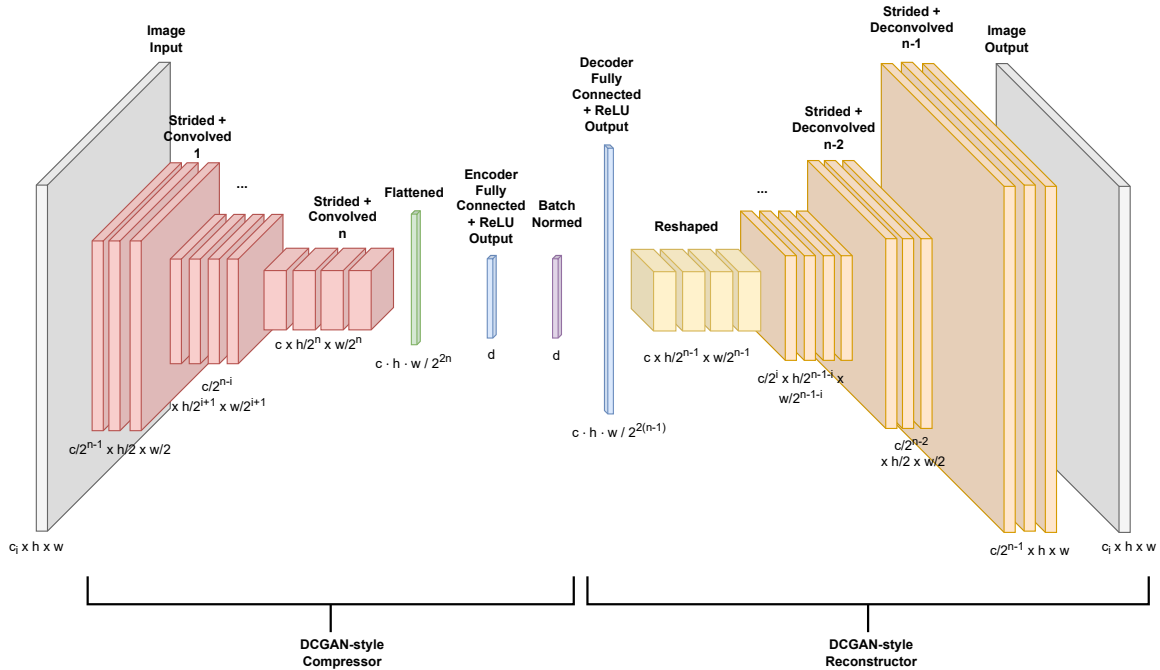


Figure 6.4: Architecture of the DCGAN-style convolutional compressor (data autoencoder's encoder) and reconstructor (data autoencoder's decoder). In the figure, c_i is the number of colour channels in the image, while c is a parameter to be chosen, dictating the number of latent channels. n dictates the number of convolutional operations to be performed. The convolutional and deconvolutional layers all have the same filter size. Note that each of the outputs of convolutional or deconvolutional layers is batch normalised and then passed through a pointwise ReLU activation function following the convolution or deconvolution operation.

6.3.2 Shared Randomness

In order to provide the transmitter and receiver with the ability to generate a probabilistic linear mapping, which is the predicted strategy for the transmitter and receiver at the saddle point of the game, we must give them a source of shared randomness. We choose to provide the networks with access to a vector of Bernoulli random variables ($p = 0.5$) as the source of shared randomness. This is concatenated to the inputs of the transmitter and receiver, respectively, with the same realisation of the random vector being supplied to both the encoder and decoder.

6.3.3 JSCC Residual Architecture

We use a fully connected architecture with residual (skip) connections between the input and output of each linear block, which includes a fully connected layer, a layer normalisation module, and a parameterised rectilinear unit (PReLU) nonlinear activation. We use a symmetric architecture for the transmitter and receiver, with the same number of layers, l , and hidden neurons n . We use a Bernoulli random vector ($p = 0.5$) as shared randomness for the encoder and decoder, with a dimension equal to the representation size, k , thus both our transmitter and receiver have input sizes $2k$, with the source randomness being concatenated to both the JSCC transmitter and receiver inputs, as shown in Figure 6.3.

For our JSCC transmitter and receiver architectures, we use two hidden layers with PReLU nonlinearities, along with one linear input layer and one linear output layer. The hidden layer sizes are chosen to be 512.

The residual architecture allows the encoder and decoder to easily produce either a probabilistic linear mapping (by passing straight through the hidden layers and using only the input and output linear mappings), or to produce a more involved nonlinear mapping, utilising the shared randomness. This straightforward architecture aids in the convergence to the saddle point of the minimax game, where the encoder and decoder mappings are probabilistic linear mappings, while retaining the ability to generate more complex mappings, if necessary, at other points in the game theoretic space during the process of competition. The added benefit of the residual blocks are that they allow the deep architecture to more effectively transmit gradients through the network in the backward pass He et al. (2016), enabling a stronger learning signal to be passed to the adversary through the Deep JSCC decoder and improved convergence. The JSCC architecture is shown in Figure 6.5, with the connectivity of the residual layers, as well as the normalisation and linear mappings.

6.3.4 Normalisation Constraints

We enforce the required constraints to satisfy the conditions for the jamming problem using a thresholded normalisation operation. We enforce a hard upper limit on the power of the transmitted representation by first subtracting the mean across batches to enforce the

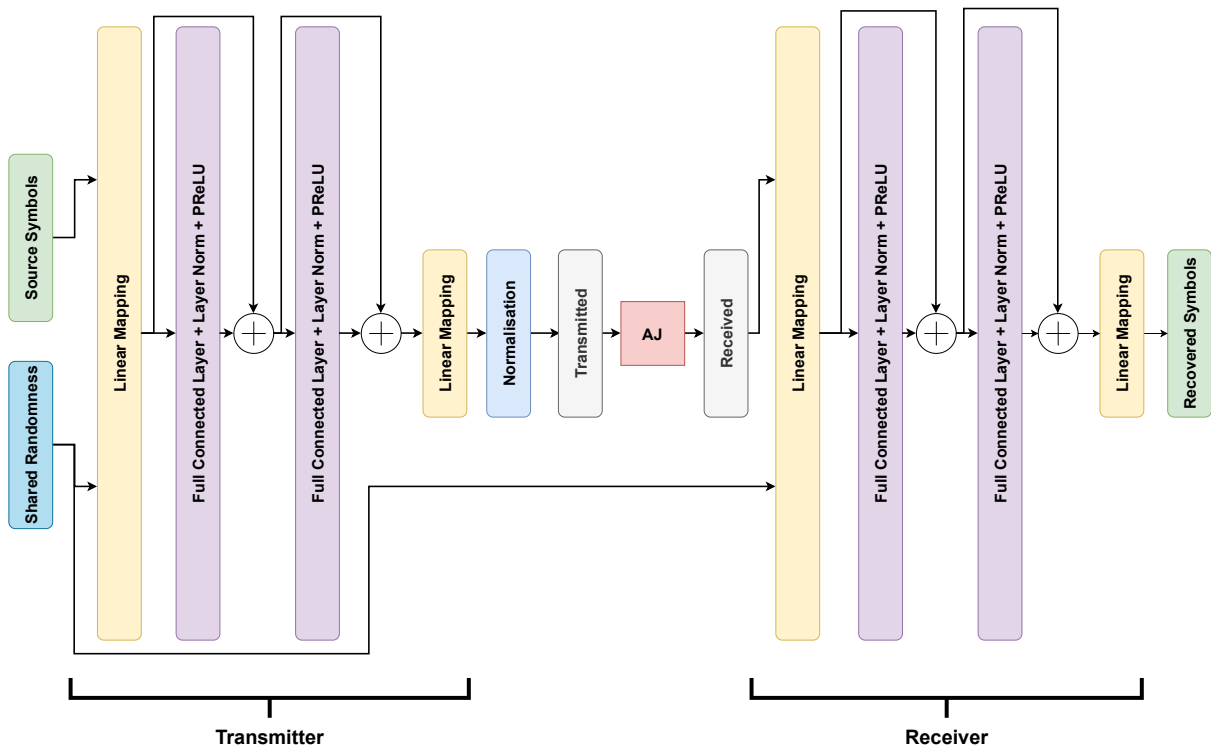


Figure 6.5: **Deep JSCC** architecture for the transmitter and receiver. We use residual, or skip, connections in order to enable the easier flow of information and gradients, allowing for the network to parametrise both a probabilistic linear mapping, which is the theoretical saddle point of the minimax communication game, as well as more complex transformations.

individual zero-mean constraints for each of the random variables in the random vector, and then normalising by the standard deviation, if it exceeds the given limit.

Given a matrix, $X \in \mathbf{R}^{b \times k}$, representing a batch of outputs to be transmitted over the channel, for either the data autoencoder's compressor, or the [Deep JSCC](#) autoencoder's transmitter, the normalisation transform is given as:

$$\begin{aligned}\mu_j &= \frac{1}{b} \sum_i X[i, j] \\ \sigma &= \sqrt{\frac{1}{b} \sum_i \sum_j (X[i, j] - \mu_j)^2} \\ \bar{X} &= \frac{X - \mu_{(0)}}{\min(\sigma, 1)}\end{aligned}\tag{6.6}$$

where $\mu_{(0)}$ represents the vector formed from the elements μ_j , where the j 'th element of the vector is μ_j , with the vector broadcast (duplicated) over the zeroth dimension b times to form a matrix of dimension $\mathbf{R}^{b \times k}$ to match the dimension of X .

We find the use of the threshold to be essential in improving the learning performance of the network. We observe that this is due to the compressor needing to produce a vector with variance that stems from processing of the data itself, rather than due to the normalisation procedure. The thresholded normalisation also implements an upper bound, rather than an equality constraint, on the empirically measured variance of the resultant representation, more closely matching the theoretical conditions for the [AJ](#) result to hold.

6.3.5 Metrics

We evaluate the performance of the technique both quantitatively and visually. For our quantitative evaluation, we make use of the [MSE](#) and the Peak Signal to Noise Ratio ([PSNR](#)) as reconstruction metrics. To measure the success of our distribution matching, we make use of two metrics: the the Fréchet Inception Distance ([FID](#)) score [Heusel et al. \(2017\)](#), measuring the realism of the generated images; and the Determinant of the Pearson Correlation Matrix ([DPC](#)) for the learned features, which is a measure of the degree of statistical independence between the learned features and thus the level of disentanglement of the features. We observe that the empirical distributions of the features appear to be marginally Gaussian, however the difficulty lies in enforcing independence between them (i.e. an identity covariance matrix - since multivariate Gaussian distributions are fully characterised by their first and second moments). Thus, we use the DPC as a single measure of the degree of independence of the features, and thus the matching of the distribution.

6.3.6 Ablation Architecture and Regularisation

We compare to two other generative autoencoder approaches: the canonical [VAE](#), using an Evidence Lower Bound ([ELBO](#)) loss term with [KL Divergence](#) distribution matching; and a [WAE](#) that uses a Maximum Mean Discrepancy ([MMD](#)) regularisation term, with an inverse multi-quadratics ([IMQ](#)) kernel.

We do not modify the data autoencoder or normalisation more than necessary, in order to ensure that experimental conditions are as consistent as possible, so that the impact of the [AJ](#) regularisation term can be accurately evaluated. In the case of the [WAE](#), we do not modify the architecture or the normalisation. In the case of the [VAE](#), due to the formulation of the [VAE](#) framework, we must remove the normalisation constraint on the output of the encoder, due to the negative impact this would have on prediction of the posterior means and standard deviations, and introduce two additional linear mappings to produce the vector of posterior means and standard deviations. These two linear mappings both map from $\mathbf{R}^k \rightarrow \mathbf{R}^k$ and are applied to the (unnormalised) output of the compressor to compute the posterior parameters.

For the [WAE](#), we use the [MMD](#) loss with a sum of [IMQ](#) kernels at a number of different scales. The scales used are $\mathcal{S} = \{.1, .2, .5, 1., 2., 5., 10.\}$ and the sum of the kernels is given by:

$$K(x, y) = \sum_{s \in \mathcal{S}} \frac{s \cdot 2k}{s \cdot 2k + \|x - y\|_2^2}$$

where k is the dimension of the latent space (e.g. 64 for CelebA) and $\|\cdot\|_2^2$ is the square L2-norm operator. These settings are chosen to be the same as those as for the original [WAE](#) formulation [Tolstikhin et al. \(2018\)](#).

For our [VAE](#) ablation experiment, we use a Gaussian encoder. We used the [DCGAN](#)-like encoder to parametrise σ_i^2 and μ_i^2 (the posterior variance and mean of the latent variable Z_i) with an additional, separate, linear readout layer to map from the encoder output to these values. Our distribution matching term is the [KL Divergence](#) from the prior, given as follows for $i \in 1 \dots k$:

$$-\frac{1}{2} \left[\sum_i \left(\log \sigma_i^2 + 1 - \sigma_i^2 - \mu_i^2 \right) \right]$$

For both ablation experiments we use the same [MSE](#) reconstruction loss as used for the [AJ](#) experiments, to allow for a fair and controlled comparison. In the case of the [VAE](#), we scale the [KL Divergence](#) term by 2.5×10^{-4} , and in the case of [WAE](#) we take the mean over batches but do not scale the distribution-matching [MMD](#) term.

6.3.7 Datasets and Optimisation Parameters

In all our experiments, we optimised both the data autoencoder - consisting of the compressor and the reconstructor - and the joint source channel coder - consisting of the transmitter and receiver - using an RMSprop optimiser, with a learning rate of 2×10^{-4} . We found that using too high an initial learning rate, especially for the channel coder, resulted in sub-optimal performance. We trained with a batch size of 128 for all experiments. For each batch, the channel coder's parameters would first be updated using the specified optimiser. We then recomputed the gradients for the updated channel coder and updated the data autoencoder's parameters. We iterated over all batches in each training dataset for a number of epochs.

We test our method on three different datasets: CelebA [Liu et al. \(2015\)](#), CIFAR-10 [Krizhevsky \(2009\)](#), and MNIST [Deng \(2012\)](#). For CelebA we train for 75 epochs with $k = 64$, for CIFAR-10 we train for 250 epochs with $k = 64$, and for MNIST we run two experiments at $k = 2$ and $k = 8$, both for 250 epochs. No image preprocessing was performed for the CIFAR-10 and MNIST experiments, but for the CelebA experiment, we performed central cropping of the images such that they were of size 140×140 pixels, followed by rescaling of the images to a size of 64×64 pixels. This is the same image preprocessing transformation applied to the CelebA dataset as used for evaluating WAEs [Tolstikhin et al. \(2018\)](#).

6.4 RESULTS

We see in Figure 6.6 sample generations from the generative model, for the three datasets, alongside results from the same model architecture trained using the WAE MMD loss, or the original VAE KL Divergence regularisation term, in place of the AJ regularisation for comparison. We see that the generations, across the datasets, are of comparable quality.

In Figure 6.7, we see a comparison of the reconstruction results across the trained networks. We see good reconstruction performance, despite the competing influence of the distribution-matching regularisation term, as compared to the ablation experiments.

In Figure 6.8, we plot a series of interpolations between points in the latent space for the network trained on each of the datasets. In each case, we randomly choose two samples from the dataset, and find the compressed representations by passing the images through the compressor network. We then linearly interpolate in the latent space, and obtain the corresponding reconstruction results.

6.5 DISCUSSION

Visually, the results in Figures 6.7 and 6.6 demonstrate performance on par with a WAE in terms of reconstruction and generation. The high DPCs in Table 6.2 suggest that our method may achieve a good level of linear feature independence, surpassing a VAE and comparable to a WAE. This is reflected in the FID scores of the models, showing that when a diagonal Gaussian prior is used to sample new data, the output samples appear comparatively realistic. However, it should be noted that the choice of hyperparameters, e.g. the regularisation constant,

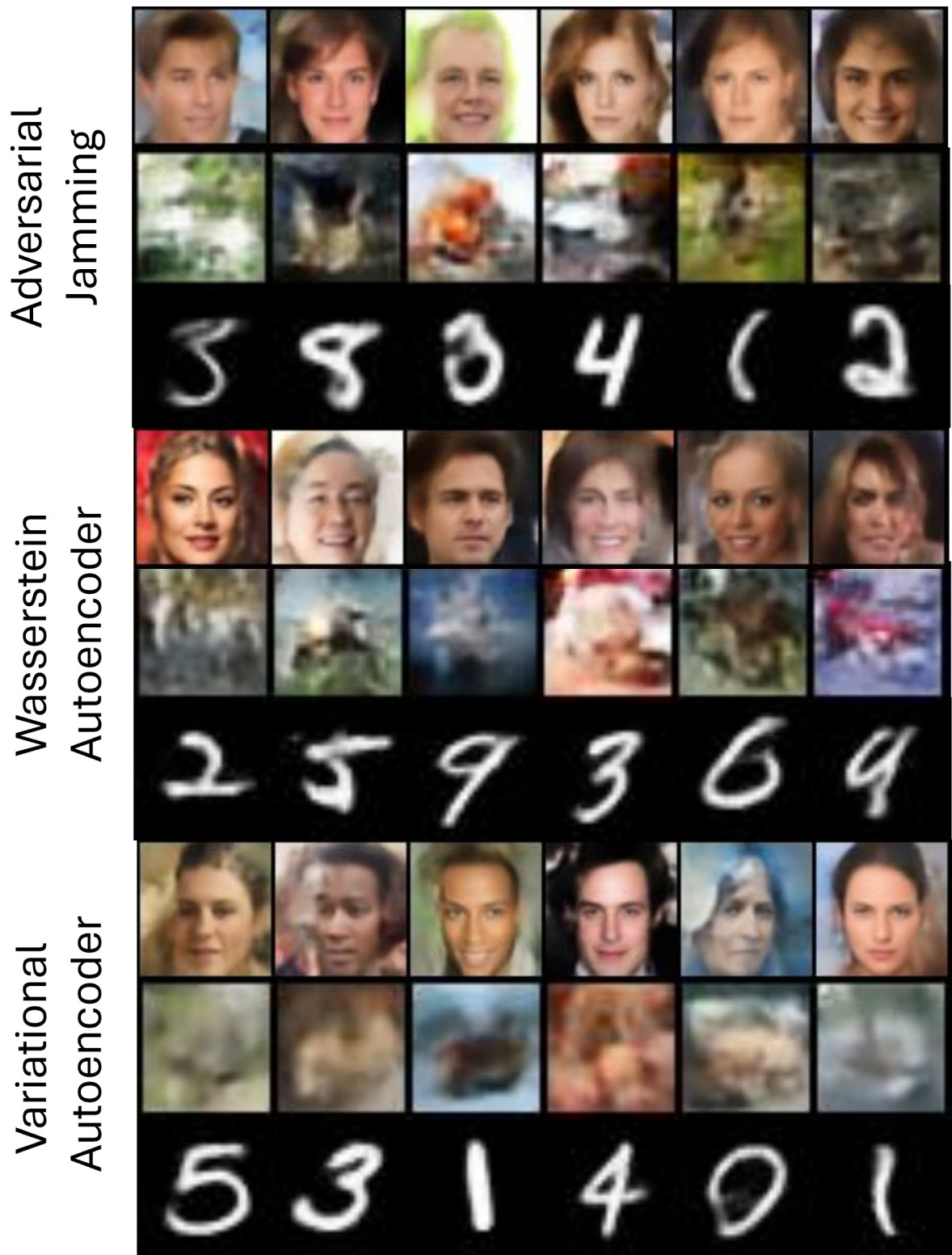


Figure 6.6: Generation results across the three datasets (top to bottom: CelebA, CIFAR-10, MNIST), for [AJ](#), as well as ablation experiments which replace the proposed [AJ](#) regularisation with a [WAE MMD](#) regularisation loss, or a [VAE](#) acKL Divergence regularisation loss. The dimension of the latent space is 8 for the MNIST results, and 64 for both the CelebA and CIFAR-10 results.



Figure 6.7: Reconstruction results across the three datasets (top to bottom: CelebA, CIFAR-10, MNIST), for [AJ](#), as well as [WAE MMD](#) and [VAE](#) ablation experiments.

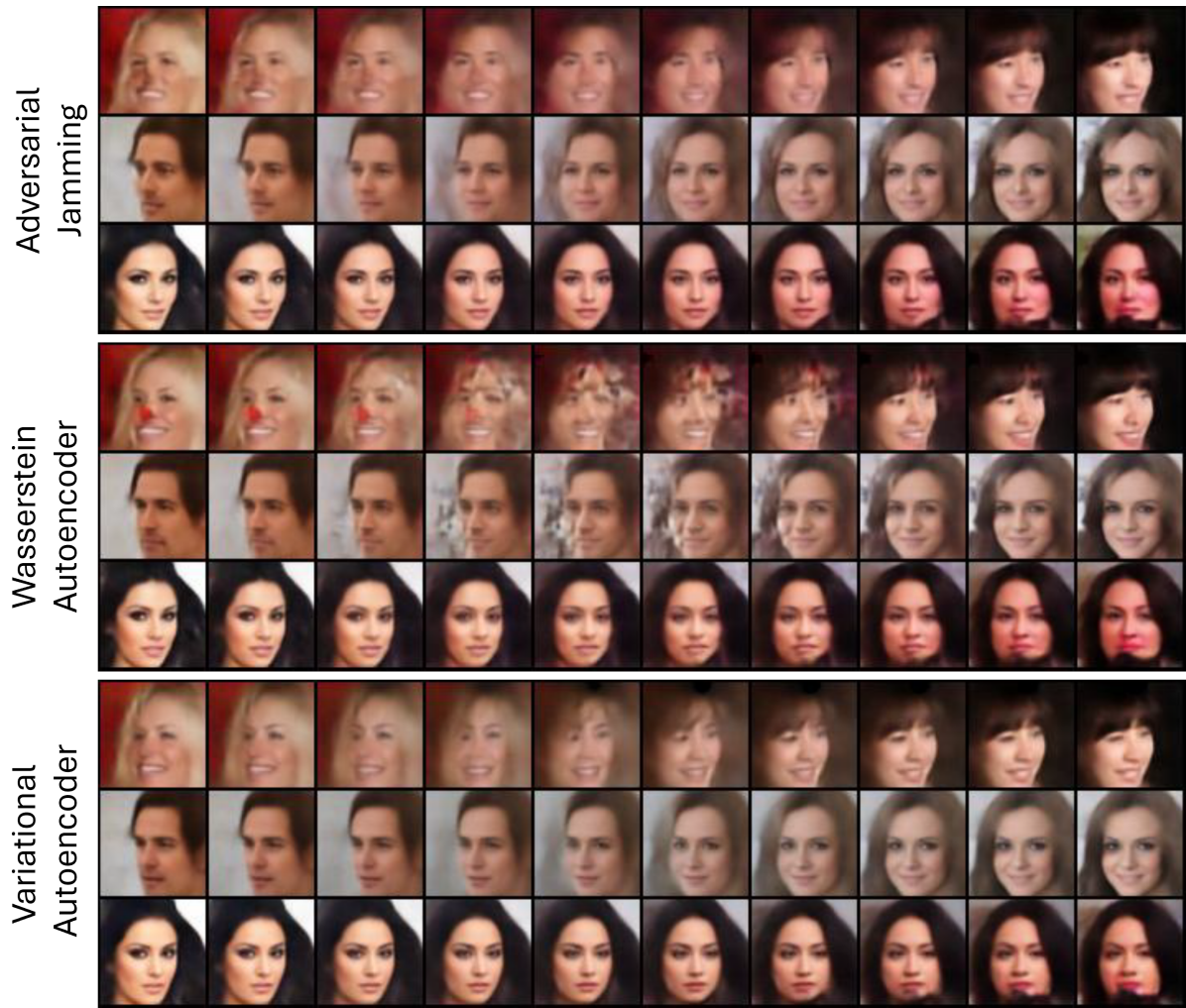


Figure 6.8: Interpolation results for the autoencoder trained using [AJ](#) as the distribution matching regularisation term for the CelebA dataset. The interpolation results demonstrate the coverage of the latent space between encoded representations. We see that the data autoencoder is able to smoothly interpolate between images in the latent space through linear interpolation of the compressed representations and corresponding reconstruction.

Table 6.2: **FID** scores, **MSE** losses for image reconstruction, and the **DPC** scores. In the case of the **FID** and the **MSE**, lower is better, whereas for the **DPC**, higher is better (maximum 1.0). These are shown for the three datasets. In each case, we show results for **AJ**, a standard **VAE**, and a **WAEs**, trained using the same parameters and architecture. We see that the performance of **AJ** is comparable to **WAEs** and outperforms **VAEs** in every case. k denotes the latent dimension (representation size). **Boldface** entries denote the best performance.

Dataset	CelebA			CIFAR-10			MNIST			MNIST		
k	64			64			2			8		
Score	FID	MSE	DPC	FID	MSE	DPC	FID	MSE	DPC	FID	MSE	DPC
AJ	45.9	0.0065	0.50	88.3	0.0105	0.57	22.2	0.0131	0.97	84.3	0.0364	1.00
WAE	44.2	0.0061	0.74	87.1	0.0105	0.75	22.0	0.0133	0.98	86.1	0.0370	1.00
VAE	50.1	0.0065	0.02	105.3	0.0116	0.17	24.1	0.0133	0.78	94.7	0.0369	0.99

may also affect the results and the trade-off between the **MSE** and **FID** (reconstruction vs. distribution matching).

Note that the **DPC** only measures the linear independence of features, so cannot be used to completely characterise the independence of the feature space, since there could be additional nonlinear dependencies. However, assuming that the features are marginally Gaussian at least, and the majority of the difficulty is in obtaining independent features, then the **DPC** is thought to be a good metric in this context. This is because a multivariate Gaussian distribution can be fully characterised by its mean vector and covariance matrix. Since the correlation matrix can be derived from the covariance matrix, we believe the **DPC** to be a good way of measuring the degree of distribution matching achieved.

As an approach to impose a prior distribution on the latent space, **AJ** opens up the possibility for novel contexts in which competition can naturally promote the matching of the distribution, with no requirement for the explicit calculation of distribution divergences. A similar approach to distribution matching, where the Jensen-Shannon divergence is implicitly minimised through an adversarial game between a generator and a discriminator network, has already achieved success in the form of **GANs** Goodfellow et al. (2014).

It is also possible for a range of other distributions to be matched based on different non-Gaussian source distributions. We have seen that in the case of input random variables with isotropic covariance matrices, the jamming noise will be a linear combination of the source distribution and the channel noise distribution. It is therefore possible for the jammer to match the source distribution arbitrarily by specifying different source distributions for the **Deep JSCC** autoencoder. This is left as future work.

Using learned representations as adversarial noise in communication-theoretic games is a new way of thinking about the representation learning problem. We note, regarding the advantages over existing methods, that the adversarial jamming approach to distribution matching offers benefits and efficiency in settings where competition is a natural part of the system - where it is a low-cost computational primitive. Such settings may include future

algorithms which make use of dynamic competition, for instance evolutionary algorithms or algorithms for the simultaneous transmission of multiple competing representations over a shared communication channel. These, again, are left as future directions to explore.

6.6 CONCLUSION

In this chapter, we have introduced [AJ](#) as a novel approach to distribution matching, offering a perspective on the use of the principles of competition and exploitation of noise for intelligent information processing. The technique implicitly imposes distribution matching by encouraging an encoder to learn a mapping from a data distribution to adversarial noise that disrupts the reconstruction task of an auxiliary joint source-channel coder. It increases the scope of possible settings in which distribution matching can be performed in the latent space of autoencoders and other neural networks. The technique has demonstrated good performance in the case of a diagonal Gaussian prior and could be extended to a range of other prior distributions, by considering different reconstruction problems involving alternative, non-Gaussian, source distributions.

CONCLUSIONS AND FUTURE WORK

In this final chapter, we aim to summarise the contributions of this thesis. Based on this, we summarise a number of limitations and challenges, and identify a number of directions for future work and on-going research. We finish this chapter with concluding remarks from the author.

7.1 SUMMARY OF CONTRIBUTIONS

Throughout this thesis, we have sought to answer the initial research question: “How can we store and process information in the presence of noise and neuronal nonlinearities?”.

We introduced the concept of neuronal computational primitives in Chapter 1, motivating the concept of neuromorphic computing, and noting that memristors could be candidates for computing of this sort, that aimed to emulate neuronal dynamics in the brain. In Chapter 2, we detailed the neuronal computational primitives and began the thesis by introducing memristive devices as a way to artificially recreate this behaviour. We demonstrated the ways in which computation in the presence of noise and nonlinearity had not only been possible using deep learning, but had indeed been accelerated by it.

We set about at starting to answer the question of how we might be able to store and process information in the presence of these nonlinearities in Chapter 3, where we highlighted the need for a model of memristive devices specifically tailored to the neuromorphic computing setting. As such, we extended existing modelling approaches in order to engineer a model which was suitable for simulation in event-based systems, since a key computational feature of neurons is their ability to process signals using spikes in time, which enables their processing of spatiotemporal data to be sparse, and hence computationally and energy efficient. We introduced the notion of the volatility state variable, which could be used to describe a range of behaviours in disparate memristive devices, allowing for a unified modelling approach. In addition, the use of this variable allowed us to explain the conditions under which a range of neuronal computational primitives could be made to come about in memristive devices, through the tuning of a few key device parameters.

In Chapter 4, we took a deeper look at one important kind of noise in memristors - resistive drift - noting that it could be connected to the neuronal computational primitive of synaptic weight decay to motivate its study. We investigated the computationally efficient modelling of this phenomenon, looking at how we could alleviate some of the need for autoregressive, recurrent modelling, by introducing delay conditioning to the model, and allowing the direct evaluation of an initial resistance and delay-conditioned distribution by formulating the modelling problem as one of communication, with the device being the noisy channel, and using generative model - a conditional Generative Adversarial Network (cGAN) - as the channel model. We demonstrated that by introducing the inductive bias of delay consistency,

through the use of an auxiliary discriminator network that we called the *delay discriminator*, we were able to allow the **cGAN** to accurately model the conditional distribution across a wide range of delays, increasing its value as a computationally efficient and differentiable model of resistive drift behaviour.

Following on from our highlighting the key consideration of energy consumption, in developing intelligent models of the world in Chapter 1, in Chapter 5, we investigated a fundamental trade-off between the storage capabilities of memristors affected by resistive drift, and the energy budget for storage. We identified a trade-off that highlighted that by increasing the energy budget allocated for information storage, one could achieve a larger capacity. We investigated this trade-off empirically, from a communication and information theoretic standpoint, by using a Deep Joint Source-Channel Coding (**Deep JSCC**) autoencoder to perform encoding and recovery of information, making use of the **cGAN** model from Chapter 4 as our channel model. We introduced the notion of Conditional Gaussian Divisive Normalisation (**CGDN**) as a technique to allow for the delay conditioning of the storage encoder and/or decoder, and demonstrated the improved performance in storage and recovery across a wide range of delays by using this conditioning technique.

Finally, in Chapter 6, we looked at one particular setting for computation in the presence of noise. Following our discussion in Section 2.6 of Chapter 2 on the benefits of noise for generalisation and robustness, we explored the use of an adversarial jamming problem as a novel technique for distribution matching in a generative modelling problem. We demonstrated that a minimax game between a transmitter and a jammer could enforce the creation of diagonal Gaussian representations as a saddle point of the game. We showed empirical results for the reconstruction, generation, and interpolation performance, from both a qualitative and quantitative perspective, showing it to be comparable in performance to other forms of distribution regularisation for generative autoencoders, and opening up a wide variety of new potential avenues for the use of noise, and the neuronal computational primitive of competition, for the learning of representations of information.

7.2 LIMITATIONS AND FUTURE WORK

There are some limitations associated with the techniques introduced in this thesis, as well as a range of avenues for future experimentation and research inspired by the contributions.

As part of the event-based modelling framework introduced in Chapter 3, we demonstrated techniques for fitting the event-based model to a drift dataset by using a constant rate approximation for the model and fitting it to a linear conductance model of the resistive switching in a titanium dioxide device. The procedure may not be as straightforward for highly nonlinear memristive devices, and thus future work could involve the collection of more extensive datasets and the exploration of more principled approaches, possibly using data-driven techniques including deep learning, to fit the model parameters to the characteristics of arbitrary memristive devices.

In Chapter 4, we explored the creation of a computationally efficient and differentiable model of memristive drift. Despite the success of our technique, we considered only the memoryless

case, while resistive drift, especially in the presence of changing volatility caused by non-constant temperature and bias, may be memory-dependent. Future research directions could include the exploration of stateful modelling of resistive drift, and indeed of the dynamics of memristive devices for non-constant bias and temperature, to build efficient, differentiable models that can be used in a variety of other storage and processing contexts.

In Chapter 5, it was shown that the autoencoder was able to store and recover information using the ground truth event-based model, following training using the cGAN approximation. Despite the ability of the autoencoder to generalise to encoding information for storage according to the dynamics of the ground truth model, in Section 5.4.4, we saw that there still existed a generalisation gap between the performance on the cGAN and the performance using the event-based model. This highlights the fact that the cGAN was not able to perfectly capture the statistics for all delay and input resistance conditions. The gap in the performance could be addressed using a number of techniques, such as improving the data normalisation transformations used for training the cGAN, or increasing the learning capacity or stability of the generative model, by using a different architecture, additional explicit or implicit inductive biases, or a different type of generative model. In addition it was noted that work could be done to extend the proposed energy trade-off to a more practical setting, taking into account uncertainty due to programming noise and additional energy expenditure associated with iterative programming algorithms.

In addition to improving the modelling accuracy, the techniques for data-driven modelling of memristive drift introduced in Chapter 5 could be extended to settings involving a range of other kinds of noise in memristors in neuromorphic applications, such as sneak-path current noise. The generative modelling principles could also be applied to other generative models, such as Variational Autoencoders (VAEs), and diffusion models, which might yield improvements in the speed or quality of model convergence. An extension could also involve the inclusion of stateful behaviour in the modelling, to enable the modelling of the volatile effects and the modelling of the change of the device state under non-constant bias and temperature conditions.

In Chapter 6, we saw the use of an adversarial jamming game to impose a diagonal Gaussian distribution on the representation of an autoencoder. Future results could include the extension of the framework to matching of different source distributions. We also saw that the performance of the model was comparable, yet no better than, Wasserstein Autoencoders (WAEs). It is possible that the complexity of the Deep JSCC architecture could affect the results, and modifications to the architecture, such as the introduction of attention, could potentially prove beneficial. Another limitation is the requirement for the auxiliary Deep JSCC architecture, which firstly introduces additional computational complexity to the training procedure, and secondly, is discarded after training, in much the same way that a discriminator network is thrown away following the training of a Generative Adversarial Network (GAN). Finding less wasteful formulations is a direction for future work. As part of the GAN formulation, Goodfellow et al. demonstrated that the competition between the discriminator and the generator resulted in implicit minimisation of the Jensen-Shannon divergence Goodfellow et al. (2014).

Inspired by this, the theoretical connections between the adversarial minimax game that was formulated and introduced, and the minimisation of certain distributional distance measures, such as f-divergences, could be an intriguing area for further exploration.

7.3 CONCLUDING REMARKS

Throughout this thesis, we have seen the notion of neuronal computational primitives come up time and time again, to inspire and improve approaches to representation learning and Artificial Intelligence (AI). The influence and the potential offered by taking inspiration from the neuronal computational primitives introduced in Section 2.1 for AI, extend far beyond the basic ideas of nonlinear activation functions. For example, we saw that competition, and adversarial competition, offer numerous benefits for AI. It is important to be able to consider these operations from an abstract perspective in order to better understand them and their potential use. Any operation which involves local averaging or normalisation can be looked at as a form of competition. In Chapter 5, we introduced the notion of Generalised Divisive Normalisation (GDN), a form of normalisation that has been observed in the human brain and has been used as a form of normalisation for image compression in artificial neural networks and in Chapter 6, we used adversarial competition to enforce distribution matching, using the notion of the worst case noise in a two-player minimax communication game. It is clear that competition is an important primitive for the learning of representations and the engineering of schemes that are robust to noise in the environment.

One could mention many more connections and insights into how computational primitives that the brain has employed for millennia have been found to be useful for engineering intelligence. An approach that uses the brain for inspiration will undoubtedly be key as we edge closer and closer to our goal. On this journey, it is clear that our continuously evolving understanding of the enigmatic human brain will continue to inspire and ignite many breakthroughs. After all, it did come up with the idea in the first place.

BIBLIOGRAPHY

- Abbey, T., Giotis, C., Serb, A., Stathopoulos, S. and Prodromakis, T. (2022), ‘Thermal Effects on Initial Volatile Response and Relaxation Dynamics of Resistive RAM Devices’, *IEEE Electron Device Letters* **43**(3), 386–389. doi: [10.1109/LED.2022.3145620](https://doi.org/10.1109/LED.2022.3145620) (pages [48](#), [74](#), [91](#)).
- Akyol, E. and Rose, K. (2013), Optimal jamming Over additive noise: Vector source-channel case, in ‘2013 51st Annual Allerton Conference on Communication, Control, and Computing (Allerton)’, pp. 1329–1335. doi: [10.1109/Allerton.2013.6736680](https://doi.org/10.1109/Allerton.2013.6736680) (pages [19](#), [55](#), [159](#), [160](#), [161](#), [162](#)).
- Akyol, E., Rose, K. and Başar, T. (2013), On Optimal Jamming Over an Additive Noise Channel, in ‘52nd IEEE Conference on Decision and Control’, pp. 3079–3084. doi: [10.1109/CDC.2013.6760352](https://doi.org/10.1109/CDC.2013.6760352) (pages [55](#), [159](#), [161](#)).
- Alexandrov, A. S., Bratkovsky, A. M., Bridle, B., Savel’ev, S. E., Strukov, D. B. and Stanley Williams, R. (2011), ‘Current-controlled negative differential resistance due to Joule heating in TiO₂’, *Applied Physics Letters* **99**(20), 202104. doi: [10.1063/1.3660229](https://doi.org/10.1063/1.3660229) (page [42](#)).
- Aoudia, F. A. and Hoydis, J. (2018), ‘End-to-End Learning of Communications Systems Without a Channel Model’, *arXiv:1804.02276 [cs, math, stat]* . (page [54](#)).
- ArC Instruments LTD (2019), ‘ArC ONE Memristor Characterisation Platform - User Manual’. (page [88](#)).
- Arimoto, S. (1972), ‘An algorithm for computing the capacity of arbitrary discrete memoryless channels’, *IEEE Transactions on Information Theory* **18**(1), 14–20. doi: [10.1109/TIT.1972.1054753](https://doi.org/10.1109/TIT.1972.1054753) (page [53](#)).
- Arjovsky, M. and Bottou, L. (2016), Towards Principled Methods for Training Generative Adversarial Networks, in ‘International Conference on Learning Representations’. (pages [58](#), [116](#)).
- Arjovsky, M., Chintala, S. and Bottou, L. (2017), Wasserstein GAN, in ‘PMLR’, pp. 214–223. (pages [58](#), [113](#)).
- Ballé, J., Laparra, V. and Simoncelli, E. P. (2016), ‘Density Modeling of Images using a Generalized Normalization Transformation’, *arXiv:1511.06281 [cs]* . (page [53](#)).
- Ballé, J., Laparra, V. and Simoncelli, E. P. (2017), ‘End-to-end Optimized Image Compression’, *arXiv:1611.01704 [cs, math]* . (page [144](#)).
- Ballé, J., Minnen, D., Singh, S., Hwang, S. J. and Johnston, N. (2018), ‘Variational image compression with a scale hyperprior’, *arXiv:1802.01436 [cs, eess, math]* . (page [53](#)).

- Basar, T. and Wu D., Y.-W. (1985), 'A complete characterization of minimax and maximin encoder- decoder policies for communication channels with incomplete statistical description', *IEEE Transactions on Information Theory* **31**(4), 482–489. doi: [10.1109/TIT.1985.1057076](https://doi.org/10.1109/TIT.1985.1057076) (page [159](#)).
- Bengio, S., Vinyals, O., Jaitly, N. and Shazeer, N. (2015), Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks, in 'Advances in Neural Information Processing Systems', Vol. 28. (page [108](#)).
- Bengio, Y., Courville, A. and Vincent, P. (2013), 'Representation Learning: A Review and New Perspectives', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **35**(8), 1798–1828. doi: [10.1109/TPAMI.2013.50](https://doi.org/10.1109/TPAMI.2013.50) (page [57](#)).
- Biolek, Z., Biolek, D. and Biolkova, V. (2012), 'Analytical Solution of Circuits Employing Voltage- and Current-Excited Memristors', *IEEE Transactions on Circuits and Systems I: Regular Papers* **59**(11), 2619–2628. doi: [10.1109/TCSI.2012.2189058](https://doi.org/10.1109/TCSI.2012.2189058) (page [48](#)).
- Bond-Taylor, S., Leach, A., Long, Y. and Willcocks, C. G. (2022), 'Deep Generative Modelling: A Comparative Review of VAEs, GANs, Normalizing Flows, Energy-Based and Autoregressive Models', *IEEE Trans Pattern Anal Mach Intell.* p. 20. (pages [56](#), [158](#)).
- Bourtsoulatze, E., Kurka, D. B. and Gunduz, D. (2019), 'Deep Joint Source-Channel Coding for Wireless Image Transmission', *IEEE Transactions on Cognitive Communications and Networking* **5**(3), 567–579. doi: [10.1109/TCCN.2019.2919300](https://doi.org/10.1109/TCCN.2019.2919300) (pages [18](#), [52](#), [53](#), [133](#), [134](#), [144](#), [146](#), [162](#)).
- Braithwaite, D. T., O'Connor, M. and Kleijn, W. B. (2020), 'Variance Constrained Autoencoding', *arXiv:2005.03807 [cs, stat]* . doi: [10.48550/arXiv.2005.03807](https://doi.org/10.48550/arXiv.2005.03807) (page [60](#)).
- Branch, M. A., Coleman, T. F. and Li, Y. (1999), 'A Subspace, Interior, and Conjugate Gradient Method for Large-Scale Bound-Constrained Minimization Problems', *SIAM Journal on Scientific Computing* **21**(1), 1–23. doi: [10.1137/S1064827595289108](https://doi.org/10.1137/S1064827595289108) (page [92](#)).
- Burg, M. F., Cadena, S. A., Denfield, G. H., Walker, E. Y., Tolia, A. S., Bethge, M. and Ecker, A. S. (2021), 'Learning divisive normalization in primary visual cortex', *PLoS Computational Biology* **17**(6), e1009028. doi: [10.1371/journal.pcbi.1009028](https://doi.org/10.1371/journal.pcbi.1009028) (page [144](#)).
- Butts, D. A., Weng, C., Jin, J., Yeh, C. I., Lesica, N. A., Alonso, J. M. and Stanley, G. B. (2007), 'Temporal precision in the neural code and the timescales of natural vision', *Nature* **449**(7158), 92–95. doi: [10.1038/nature06105](https://doi.org/10.1038/nature06105) (page [37](#)).
- Carbajal, J. P., Dambre, J., Hermans, M. and Schrauwen, B. (2015), 'Memristor Models for Machine Learning', *Neural Computation* **27**(3), 725–747. doi: [10.1162/NECO_a_00694](https://doi.org/10.1162/NECO_a_00694) (pages [38](#), [40](#), [45](#), [48](#)).
- Chacron, M. J., Longtin, A. and Maler, L. (2004), 'To Burst or Not to Burst?', *Journal of Computational Neuroscience* **17**(2), 127–136. doi: [10.1023/B:JCNS.0000037677.58916.6b](https://doi.org/10.1023/B:JCNS.0000037677.58916.6b) (page [37](#)).

- Chang, T., Jo, S.-H. and Lu, W. (2011), ‘Short-Term Memory to Long-Term Memory Transition in a Nanoscale Memristor’, *ACS Nano* 5(9), 7669–7676. doi: [10.1021/nn202983n](https://doi.org/10.1021/nn202983n) (page 48).
- Chen, T., Kornblith, S., Norouzi, M. and Hinton, G. (2020), A simple framework for contrastive learning of visual representations, in ‘Proceedings of the 37th International Conference on Machine Learning’, Vol. 119 of *ICML’20*, JMLR.org, pp. 1597–1607. (page 25).
- Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H. and Bengio, Y. (2014), Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation, in A. Moschitti, B. Pang and W. Daelemans, eds, ‘Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)’, Association for Computational Linguistics, pp. 1724–1734. doi: [10.3115/v1/D14-1179](https://doi.org/10.3115/v1/D14-1179) (page 108).
- Chua, L. (1971), ‘Memristor -The missing circuit element’, *IEEE Transactions on Circuit Theory* 18(5), 507–519. doi: [10.1109/TCT.1971.1083337](https://doi.org/10.1109/TCT.1971.1083337) (pages 39, 67).
- Chua, L. (2011), ‘Resistance switching memories are memristors’, *Applied Physics A* 102(4), 765–783. doi: [10.1007/s00339-011-6264-9](https://doi.org/10.1007/s00339-011-6264-9) (pages 40, 67).
- Chua, L. and Sung Mo Kang (1976), ‘Memristive devices and systems’, *Proceedings of the IEEE* 64(2), 209–223. doi: [10.1109/PROC.1976.10092](https://doi.org/10.1109/PROC.1976.10092) (pages 40, 47, 68).
- Cover, T. M. and Thomas, J. A. (2006), *Elements of Information Theory*, 2nd edn, Wiley-Interscience, Hoboken, N.J. (page 54).
- Cybenko, G. (1989), ‘Approximation by superpositions of a sigmoidal function’, *Mathematics of Control, Signals and Systems* 2(4), 303–314. doi: [10.1007/BF02551274](https://doi.org/10.1007/BF02551274) (page 37).
- Dauphin, Y. N., Fan, A., Auli, M. and Grangier, D. (2017), ‘Language Modeling with Gated Convolutional Networks’, *arXiv:1612.08083 [cs]* . doi: [10.48550/arXiv.1612.08083](https://doi.org/10.48550/arXiv.1612.08083) (page 117).
- Dayan, P. and Abbott, L. F. (2001), *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*, Massachusetts Institute of Technology Press. (pages 37, 100).
- De Vries, B. and Principe, J. C. (1992), ‘The gamma model—A new neural model for temporal processing’, *Neural Networks* 5(4), 565–576. doi: [10.1016/S0893-6080\(05\)80035-8](https://doi.org/10.1016/S0893-6080(05)80035-8) (page 109).
- de Vries, H., Strub, F., Mary, J., Larochelle, H., Pietquin, O. and Courville, A. (2017), Modulating early visual processing by language, in ‘Proceedings of the 31st International Conference on Neural Information Processing Systems’, NIPS’17, Curran Associates Inc., pp. 6597–6607. (page 144).
- Deng, L. (2012), ‘The MNIST Database of Handwritten Digit Images for Machine Learning Research’, *IEEE Signal Processing Magazine* 29(6), 141–142. (page 169).
- Diggavi, S. and Cover, T. (2001), ‘The worst additive noise under a covariance constraint’, *IEEE Transactions on Information Theory* 47(7), 3072–3081. doi: [10.1109/18.959289](https://doi.org/10.1109/18.959289) (page 55).

- Ding, X., Wang, Y., Xu, Z., Welch, W. J. and Wang, Z. J. (2022), CcGAN: Continuous Conditional Generative Adversarial Networks for Image Generation, in 'International Conference on Learning Representations'. (pages [54](#), [59](#), [107](#)).
- Ding, X., Wang, Y., Xu, Z., Welch, W. J. and Wang, Z. J. (2023), 'Continuous Conditional Generative Adversarial Networks: Novel Empirical Losses and Label Input Mechanisms', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **45**(7), 8143–8158. doi: [10.1109/TPAMI.2022.3228915](#) (pages [54](#), [59](#)).
- Dinh, L., Sohl-Dickstein, J. and Bengio, S. (2016), Density estimation using Real NVP, in 'International Conference on Learning Representations'. (pages [57](#), [158](#)).
- El-Geresy, W., Papavassiliou, C. and Gündüz, D. (2024a), 'Resistive Drift of Titanium Dioxide Memristor Devices', *IEEE DataPort* . doi: [10.21227/4qbk-zh17](#) (page [88](#)).
- El-Geresy, W., Papavassiliou, C. and Gündüz, D. (2024b), 'Simulated Resistive Drift of Memristive Devices', *IEEE DataPort* . doi: [10.21227/edzz-7d60](#) (page [109](#)).
- Engel, J. H., Eryilmaz, S. B., Kim, S., BrightSky, M., Lam, C., Lung, H. L., Olshausen, B. A. and Wong, H. S. (2015), Capacity optimization of emerging memory systems: A shannon-inspired approach to device characterization, in '2014 IEEE International Electron Devices Meeting', IEEE. doi: [10.1109/IEDM.2014.7047134](#) (page [53](#)).
- Engel, J. H., Eryilmaz, S. B., Kim, S., BrightSky, M., Lam, C., Lung, H.-L., Olshausen, B. A. and Wong, H.-S. P. (2017), 'Opportunities for Analog Coding in Emerging Memory Systems', *arXiv:1701.06063 [cs, math]* . doi: [10.48550/arXiv.1701.06063](#) (pages [43](#), [44](#), [49](#), [53](#)).
- Fil, M., Mesinovic, M., Morris, M. and Wildberger, J. (2021), 'Beta-VAE Reproducibility: Challenges and Extensions', *arXiv:2112.14278 [cs]* . doi: [10.48550/arXiv.2112.14278](#) (page [58](#)).
- FitzHugh, R. (1961), 'Impulses and Physiological States in Theoretical Models of Nerve Membrane', *Biophysical Journal* **1**(6), 445–466. (page [36](#)).
- Friston, K. (2010), 'The free-energy principle: A unified brain theory?', *Nature Reviews Neuroscience* **11**(2), 127–138. doi: [10.1038/nrn2787](#) (page [26](#)).
- Friston, K., Kilner, J. and Harrison, L. (2006), 'A free energy principle for the brain', *Journal of Physiology Paris* **100**(1-3), 70–87. doi: [10.1016/j.jphysparis.2006.10.001](#) (page [26](#)).
- Fu, T., Liu, X., Gao, H., Ward, J. E., Liu, X., Yin, B., Wang, Z., Zhuo, Y., Walker, D. J. F., Joshua Yang, J., Chen, J., Lovley, D. R. and Yao, J. (2020), 'Bioinspired bio-voltage memristors', *Nature Communications* **11**(1), 1861. doi: [10.1038/s41467-020-15759-y](#) (pages [100](#), [133](#), [137](#)).
- Gal, Y. and Ghahramani, Z. (2016), Dropout as a Bayesian approximation: Representing model uncertainty in deep learning, in 'Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48', ICML'16, JMLR.org, pp. 1050–1059. (page [59](#)).

- Gallo, M. L. and Sebastian, A. (2020), 'An overview of phase-change memory device physics', *Journal of Physics D: Applied Physics* **53**(21), 213002. doi: [10.1088/1361-6463/ab7794](https://doi.org/10.1088/1361-6463/ab7794) (pages [43](#), [75](#)).
- Gao, L., Ren, Q., Sun, J., Han, S.-T. and Zhou, Y. (2021), 'Memristor modeling: Challenges in theories, simulations, and device variability', *Journal of Materials Chemistry C* **9**(47), 16859–16884. doi: [10.1039/D1TC04201G](https://doi.org/10.1039/D1TC04201G) (page [46](#)).
- Georgopoulos, A. P., Schwartz, A. B. and Kettner, R. E. (1986), 'Neuronal population coding of movement direction', *Science (New York, N.Y.)* **233**(4771), 1416–1419. doi: [10.1126/science.3749885](https://doi.org/10.1126/science.3749885) (page [37](#)).
- Gerstner, Wulfram. and Kistler, W. M. (2002), *Spiking Neuron Models : Single Neurons, Populations, Plasticity*, Cambridge University Press, Cambridge. (page [36](#)).
- Gerstner, W., Kistler, W. M., Naud, R. and Paninski, L. (2014), *Neuronal Dynamics: From Single Neurons to Networks and Models of Cognition*, Cambridge University Press. doi: [10.1017/CBO9781107447615](https://doi.org/10.1017/CBO9781107447615) (page [100](#)).
- Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y. (2014), 'Generative Adversarial Networks', *Advances in neural information processing systems* . (pages [39](#), [56](#), [58](#), [107](#), [113](#), [158](#), [159](#), [173](#), [177](#)).
- Goodfellow, I. J., Shlens, J. and Szegedy, C. (2015), Explaining and Harnessing Adversarial Examples, in Y. Bengio and Y. LeCun, eds, '3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings'. (page [59](#)).
- Guo, Q., Gu, W. and Tao, Z. (2015), Properties of Quadratic Flux-Controlled and Charge-Controlled Memristor, in 'International Conference on Advances in Mechanical Engineering and Industrial Informatics (AMEII 2015)', Atlantis Press, pp. 1461–1465. doi: [10.2991/ameii-15.2015.269](https://doi.org/10.2991/ameii-15.2015.269) (page [67](#)).
- Haarnoja, T., Zhou, A., Abbeel, P. and Levine, S. (2018), Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor, in 'Proceedings of the 35th International Conference on Machine Learning', PMLR, pp. 1861–1870. (page [59](#)).
- Han, J.-H., Kushner, S. A., Yiu, A. P., Cole, C. J., Matynia, A., Brown, R. A., Neve, R. L., Guzowski, J. F., Silva, A. J. and Josselyn, S. A. (2007), 'Neuronal Competition and Selection During Memory Formation', *Science* **316**(5823), 457–460. doi: [10.1126/science.1139438](https://doi.org/10.1126/science.1139438) (page [39](#)).
- He, K., Zhang, X., Ren, S. and Sun, J. (2016), Deep Residual Learning for Image Recognition, in '2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)', pp. 770–778. doi: [10.1109/CVPR.2016.90](https://doi.org/10.1109/CVPR.2016.90) (pages [114](#), [165](#)).
- Hebb, D. O. (1949), *The Organization of Behavior; a Neuropsychological Theory*, The Organization of Behavior; a Neuropsychological Theory, Wiley, Oxford, England. (pages [38](#), [100](#)).

- Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B. and Hochreiter, S. (2017), GANs trained by a two time-scale update rule converge to a local nash equilibrium, in 'Advances in Neural Information Processing Systems', Vol. 30. (page 167).
- Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S. and Lerchner, A. (2017), ' β -VAE: Learning Basic Visual Concepts with a Constrained Variational Framework', *International Conference on Learning Representations* . (page 58).
- Hinton, G. E. and Sejnowski, T. J. (1983), 'Optimal Perceptual Inference', *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition* . (page 57).
- Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R. R. (2012), 'Improving neural networks by preventing co-adaptation of feature detectors', *arXiv:1207.0580 [cs]* . doi: 10.48550/arXiv.1207.0580 (page 59).
- Ho, J., Jain, A. and Abbeel, P. (2020), Denoising Diffusion Probabilistic Models, in 'Advances in Neural Information Processing Systems', Vol. 33, Curran Associates, Inc., pp. 6840–6851. (pages 25, 158).
- Hochreiter, S. and Schmidhuber, J. (1997), 'Long Short-Term Memory', *Neural Computation* 9(8), 1735–1780. doi: 10.1162/neco.1997.9.8.1735 (page 108).
- Hodgkin, A. L. and Huxley, A. F. (1952), 'A quantitative description of membrane current and its application to conduction and excitation in nerve', *The Journal of Physiology* 117(4), 500–544. doi: 10.1113/jphysiol.1952.sp004764 (pages 36, 100).
- Hornik, K., Stinchcombe, M. and White, H. (1989), 'Multilayer feedforward networks are universal approximators', *Neural Networks* . doi: 10.1016/0893-6080(89)90020-8 (page 37).
- Huszár, F. (2015), 'How (not) to Train your Generative Model: Scheduled Sampling, Likelihood, Adversary?', *arXiv:1511.05101 [cs, math, stat]* . (page 108).
- Ielmini, D. (2011), 'Modeling the Universal Set/Reset Characteristics of Bipolar RRAM by Field- and Temperature-Driven Filament Growth', *IEEE Transactions on Electron Devices* 58(12), 4309–4317. doi: 10.1109/TED.2011.2167513 (pages 133, 137).
- Ielmini, D., Lavizzari, S., Sharma, D. and Lacaita, A. L. (2007), Physical interpretation, modeling and impact on phase change memory (PCM) reliability of resistance drift due to chalcogenide structural relaxation, in '2007 IEEE International Electron Devices Meeting', pp. 939–942. doi: 10.1109/IEDM.2007.4419107 (page 48).
- Ielmini, D., Nardi, F., Cagli, C. and Lacaita, A. L. (2010), 'Size-Dependent Retention Time in NiO-Based Resistive-Switching Memories', *IEEE Electron Device Letters* 31(4), 353–355. doi: 10.1109/LED.2010.2040799 (pages 133, 137).
- Iranmehr, E., Shouraki, S. B., Faraji, M. M., Bagheri, N. and Linares-Barranco, B. (2019), 'Bio-Inspired Evolutionary Model of Spiking Neural Networks in Ionic Liquid Space', *Frontiers in Neuroscience* 13, 1085. doi: 10.3389/fnins.2019.01085 (page 100).

- Isik, B., Choi, K., Zheng, X., Weissman, T., Ermon, S., Wong, H.-S. P. and Alaghi, A. (2021), 'Neural Network Compression for Noisy Storage Devices', *arXiv:2102.07725 [cs]* . (page 54).
- Izhikevich, E. (2003), 'Simple model of spiking neurons', *IEEE Transactions on Neural Networks* **14**(6), 1569–1572. doi: [10.1109/TNN.2003.820440](https://doi.org/10.1109/TNN.2003.820440) (page 36).
- Izhikevich, E. (2004), 'Which Model to Use for Cortical Spiking Neurons?', *IEEE Transactions on Neural Networks* **15**(5), 1063–1070. doi: [10.1109/TNN.2004.832719](https://doi.org/10.1109/TNN.2004.832719) (page 37).
- Jeong, D. S., Schroeder, H. and Waser, R. (2007), 'Coexistence of Bipolar and Unipolar Resistive Switching Behaviors in a Pt/TiO₂/Pt Stack', *Electrochemical and Solid-State Letters* **10**(8), G51. doi: [10.1149/1.2742989](https://doi.org/10.1149/1.2742989) (page 42).
- Jiang, Z., Chen, T., Chen, T. and Wang, Z. (2020), Robust Pre-Training by Adversarial Contrastive Learning, in 'Advances in Neural Information Processing Systems', Vol. 33, Curran Associates, Inc., pp. 16199–16210. (page 159).
- Jo, S. H., Kim, K.-H. and Lu, W. (2009), 'Programmable Resistance Switching in Nanoscale Two-Terminal Devices', *Nano Letters* **9**(1), 496–500. doi: [10.1021/nl803669s](https://doi.org/10.1021/nl803669s) (pages 47, 48).
- Joksas, D., Wang, E., Barmptsalos, N., Ng, W. H., Kenyon, A. J., Constantinides, G. A. and Mehonic, A. (2022), 'Nonideality-Aware Training for Accurate and Robust Low-Power Memristive Neural Networks', *Advanced Science* **9**(17), 2105784. doi: [10.1002/advs.202105784](https://doi.org/10.1002/advs.202105784) (page 54).
- Jose, S. T. and Kulkarni, A. A. (2020), 'Shannon Meets von Neumann: A Minimax Theorem for Channel Coding in the Presence of a Jammer', *IEEE Transactions on Information Theory* **66**(5), 2842–2859. doi: [10.1109/TIT.2020.2971682](https://doi.org/10.1109/TIT.2020.2971682) (page 54).
- Kandel, E. R., ed. (2013), *Principles of Neural Science*, 5th ed edn, McGraw-Hill, New York. (page 100).
- Kim, M., Tack, J. and Hwang, S. J. (2020), 'Adversarial Self-Supervised Contrastive Learning', *Advances in Neural Information Processing Systems* **33**, 2983–2994. (page 159).
- Kim, T., Hu, S., Kim, J., Kwak, J. Y., Park, J., Lee, S., Kim, I., Park, J.-K. and Jeong, Y. (2021), 'Spiking Neural Network (SNN) With Memristor Synapses Having Non-linear Weight Update', *Frontiers in Computational Neuroscience* **15**. (page 45).
- Kingma, D. P. and Ba, J. (2017), 'Adam: A Method for Stochastic Optimization', *arXiv:1412.6980 [cs]* . doi: [10.48550/arXiv.1412.6980](https://doi.org/10.48550/arXiv.1412.6980) (pages 119, 147).
- Kingma, D. P. and Welling, M. (2014), Auto-Encoding Variational Bayes, in '2nd International Conference on Learning Representations'. (pages 32, 57, 158).
- Kolouri, S., Pope, P. E., Martin, C. E. and Rohde, G. K. (2018), 'Sliced-Wasserstein Autoencoder: An Embarrassingly Simple Generative Model', *arXiv:1804.01947 [cs, stat]* . doi: [10.48550/arXiv.1804.01947](https://doi.org/10.48550/arXiv.1804.01947) (page 58).

- Krizhevsky, A. (2009), ‘Learning Multiple Layers of Features from Tiny Images’, *Technical Report, Department of Computer Science, University of Toronto* . (pages 147, 169).
- Kurka, D. B. and Gunduz, D. (2019), ‘Successive Refinement of Images with Deep Joint Source-Channel Coding’, *arXiv:1903.06333 [cs, math]* . (page 52).
- Kurka, D. B. and Gündüz, D. (2020), ‘DeepJSCC-f: Deep Joint-Source Channel Coding of Images with Feedback’, *IEEE Journal on Selected Areas in Information Theory* 1(1), 178–193. (page 52).
- Kwon, D.-H., Kim, K. M., Jang, J. H., Jeon, J. M., Lee, M. H., Kim, G. H., Li, X.-S., Park, G.-S., Lee, B., Han, S., Kim, M. and Hwang, C. S. (2010), ‘Atomic structure of conducting nanofilaments in TiO₂ resistive switching memory.’, *Nature nanotechnology* . doi: 10.1038/nnano.2009.456 (page 42).
- Lapidoth, A. (1996), ‘Nearest neighbor decoding for additive non-Gaussian noise channels’, *IEEE Transactions on Information Theory* 42(5), 1520–1529. doi: 10.1109/18.532892 (page 55).
- Lee, J. (1999), ‘Combined source/channel coding and a new coding scheme for data storage systems’, *IEEE Transactions on Magnetics* 35(5), 4372–4377. doi: 10.1109/20.799088 (page 53).
- Lee, K., Chang, H., Jiang, L., Zhang, H., Tu, Z. and Liu, C. (2024), ‘ViTGAN: Training GANs with Vision Transformers’, *arXiv:2107.04589 [cs, eess]* . doi: 10.48550/arXiv.2107.04589 (page 58).
- Li, J., Luan, B. and Lam, C. (2012), Resistance drift in phase change memory, in ‘2012 IEEE International Reliability Physics Symposium (IRPS)’, pp. 6C.1.1–6C.1.6. doi: 10.1109/IRPS.2012.6241871 (pages 43, 48).
- Lin, Z., Khetan, A., Fanti, G. and Oh, S. (2020), ‘PacGAN: The Power of Two Samples in Generative Adversarial Networks’, *IEEE Journal on Selected Areas in Information Theory* 1(1), 324–335. doi: 10.1109/JSAIT.2020.2983071 (pages 58, 117).
- Liu, Z., Luo, P., Wang, X. and Tang, X. (2015), Deep Learning Face Attributes in the Wild, in ‘2015 IEEE International Conference on Computer Vision (ICCV)’, pp. 3730–3738. doi: 10.1109/ICCV.2015.425 (page 169).
- Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I. and Frey, B. (2016), ‘Adversarial Autoencoders’, *arXiv:1511.05644 [cs]* . doi: 10.48550/arXiv.1511.05644 (pages 57, 58).
- Malik, A., Papavassiliou, C. and Stathopoulos, S. (2021), A Stochastic Compact Model Describing Memristor Plasticity and Volatility, in ‘2021 28th IEEE International Conference on Electronics, Circuits, and Systems (ICECS)’, IEEE, pp. 1–4. doi: 10.1109/ICECS53924.2021.9665591 (page 64).
- Malik, A., Papavassiliou, C. and Stathopoulos, S. (2022), An Absorbing Markov Chain Model for Stochastic Memristive Devices, in ‘2022 11th International Conference on Modern Circuits and Systems Technologies (MOCASST)’, pp. 1–4. doi: 10.1109/MOCASST54814.2022.9837672 (pages 47, 48).

- Mandic, D. P. and Chambers, J. A. (2001), Introduction, in 'Recurrent Neural Networks for Prediction', John Wiley & Sons, Ltd, chapter 1, pp. 1–8. doi: [10.1002/047084535X.ch1](https://doi.org/10.1002/047084535X.ch1) (page [37](#)).
- Mantegazza, D., Ielmini, D., Pirovano, A. and Lacaita, A. L. (2010), 'Incomplete Filament Crystallization During Set Operation in PCM Cells', *IEEE Electron Device Letters* **31**(4), 341–343. doi: [10.1109/LED.2010.2042273](https://doi.org/10.1109/LED.2010.2042273) (pages [44](#), [75](#)).
- Marino, J. (2022), 'Predictive Coding, Variational Autoencoders, and Biological Connections', *Neural Computation* **34**(1), 1–44. doi: [10.1162/neco_a_01458](https://doi.org/10.1162/neco_a_01458) (page [26](#)).
- McFadden, J. A. (1965), 'The Entropy of a Point Process', *Journal of the Society for Industrial and Applied Mathematics* **13**(4), 988–994. (page [37](#)).
- Mead, C. (1990), 'Neuromorphic electronic systems', *Proceedings of the IEEE* **78**(10), 1629–1636. doi: [10.1109/5.58356](https://doi.org/10.1109/5.58356) (pages [29](#), [62](#)).
- Mead, C., Ismail, M. and Allen, J., eds (1989), *Analog VLSI Implementation of Neural Systems*, Vol. 80 of *The Kluwer International Series in Engineering and Computer Science*, Springer US, Boston, MA. doi: [10.1007/978-1-4613-1639-8](https://doi.org/10.1007/978-1-4613-1639-8) (page [29](#)).
- Mehonic, A., Kenyon, A., Sebastian, A., Rajendran, B., Simeone, O. and Vasilaki, E. (2020), 'Memristors - From in-memory computing, deep learning acceleration, spiking neural networks, to the future of neuromorphic and bio-inspired computing', *Advanced Intelligent Systems* **2**(11). doi: [10.1002/aisy.202000085](https://doi.org/10.1002/aisy.202000085) (page [45](#)).
- Mehonic, A. and Kenyon, A. J. (2016), 'Emulating the Electrical Activity of the Neuron Using a Silicon Oxide RRAM Cell', *Frontiers in Neuroscience* **10**. doi: [10.3389/fnins.2016.00057](https://doi.org/10.3389/fnins.2016.00057) (page [45](#)).
- Mirza, M. and Osindero, S. (2014), 'Conditional Generative Adversarial Nets', *arXiv:1411.1784 [cs, stat]*. doi: [10.48550/arXiv.1411.1784](https://doi.org/10.48550/arXiv.1411.1784) (page [107](#)).
- Molter, T. W. and Nugent, M. A. (2016), The Generalized Metastable Switch Memristor Model, in 'CNNA 2016; 15th International Workshop on Cellular Nanoscale Networks and Their Applications', pp. 1–2. (pages [31](#), [47](#), [48](#), [61](#), [63](#), [64](#), [66](#), [67](#), [70](#)).
- Montemurro, M. A., Rasch, M. J., Murayama, Y., Logothetis, N. K. and Panzeri, S. (2008), 'Phase-of-Firing Coding of Natural Visual Stimuli in Primary Visual Cortex', *Current Biology* **18**(5), 375–380. doi: [10.1016/j.cub.2008.02.023](https://doi.org/10.1016/j.cub.2008.02.023) (page [37](#)).
- Nandakumar, S. R., Boybat, I., Joshi, V., Piveteau, C., Le Gallo, M., Rajendran, B., Sebastian, A. and Eleftheriou, E. (2019), Phase-Change Memory Models for Deep Learning Training and Inference, in '2019 26th IEEE International Conference on Electronics, Circuits and Systems (ICECS)', pp. 727–730. doi: [10.1109/ICECS46596.2019.8964852](https://doi.org/10.1109/ICECS46596.2019.8964852) (page [44](#)).
- Naous, R., Al-Shedivat, M. and Salama, K. N. (2016), 'Stochasticity Modeling in Memristors', *IEEE Transactions on Nanotechnology* **15**(1), 15–28. doi: [10.1109/TNANO.2015.2493960](https://doi.org/10.1109/TNANO.2015.2493960) (page [64](#)).

- Nguyen, H. T., Bottone, S., Kim, K. T., Chiang, M. and Poor, H. V. (2021), Adversarial Neural Networks for Error Correcting Codes, *in* '2021 IEEE Global Communications Conference (GLOBECOM)', IEEE Press, pp. 01–06. doi: [10.1109/GLOBECOM46510.2021.9685472](https://doi.org/10.1109/GLOBECOM46510.2021.9685472) (page [159](#)).
- Nigus, M., Priyadarshini, R. and Mehra, R. M. (2019), 'Stochastic and novel generic scalable window function-based deterministic memristor SPICE model comparison and implementation for synaptic circuit design', *SN Applied Sciences* **2**(1), 128. doi: [10.1007/s42452-019-1888-z](https://doi.org/10.1007/s42452-019-1888-z) (page [43](#)).
- Nugent, M. A. and Molter, T. W. (2014), 'AHaH Computing-From Metastable Switches to Attractors to Machine Learning', *PLoS ONE* **9**(2), e85175. doi: [10.1371/journal.pone.0085175](https://doi.org/10.1371/journal.pone.0085175) (pages [31](#), [48](#), [63](#)).
- Oh, S., Huang, Z., Shi, Y. and Kuzum, D. (2019), 'The Impact of Resistance Drift of Phase Change Memory (PCM) Synaptic Devices on Artificial Neural Network Performance', *IEEE Electron Device Letters* **40**(8), 1325–1328. doi: [10.1109/LED.2019.2925832](https://doi.org/10.1109/LED.2019.2925832) (page [45](#)).
- O'Shea, T. J., Roy, T. and West, N. (2019), Approximating the Void: Learning Stochastic Channel Models from Observation with Variational Generative Adversarial Networks, *in* '2019 International Conference on Computing, Networking and Communications, ICNC 2019', Institute of Electrical and Electronics Engineers Inc., pp. 681–686. doi: [10.1109/ICCNC.2019.8685573](https://doi.org/10.1109/ICCNC.2019.8685573) (page [54](#)).
- O'Shea, T. J., Roy, T., West, N. and Hilburn, B. C. (2018), Physical Layer Communications System Design Over-the-Air Using Adversarial Networks, *in* '2018 26th European Signal Processing Conference (EUSIPCO)', IEEE, pp. 529–532. doi: [10.23919/EUSIPCO.2018.8553233](https://doi.org/10.23919/EUSIPCO.2018.8553233) (page [159](#)).
- Pickett, M. D., Strukov, D. B., Borghetti, J. L., Yang, J. J., Snider, G. S., Stewart, D. R. and Williams, R. S. (2009), 'Switching dynamics in titanium dioxide memristive devices', *Journal of Applied Physics* **106**(7), 074508. doi: [10.1063/1.3236506](https://doi.org/10.1063/1.3236506) (pages [43](#), [47](#), [67](#), [70](#)).
- Prodromakis, T., Peh, B. P., Papavassiliou, C. and Toumazou, C. (2011), 'A Versatile Memristor Model With Nonlinear Dopant Kinetics', *IEEE Transactions on Electron Devices* **58**(9), 3099–3105. doi: [10.1109/TED.2011.2158004](https://doi.org/10.1109/TED.2011.2158004) (page [47](#)).
- Purves, D., ed. (2018), *Neuroscience*, 6th ed edn, Oxford University Press, New York. (pages [36](#), [38](#), [39](#), [157](#)).
- Qin, M. and Vucinic, D. (2018), 'Noisy Computations during Inference: Harmful or Helpful?', *arXiv:1811.10649 [cs, stat]* . (pages [45](#), [54](#), [59](#)).
- Radford, A., Metz, L. and Chintala, S. (2016), 'Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks', *arXiv:1511.06434 [cs]* . doi: [10.48550/arXiv.1511.06434](https://doi.org/10.48550/arXiv.1511.06434) (page [164](#)).

- Robbins, H. and Monro, S. (1951), 'A Stochastic Approximation Method', *The Annals of Mathematical Statistics* **22**(3), 400–407. doi: [10.1214/aoms/1177729586](https://doi.org/10.1214/aoms/1177729586) (page 59).
- Roldán, J. B., González-Cordero, G., Picos, R., Miranda, E., Palumbo, F., Jiménez-Molinos, F., Moreno, E., Maldonado, D., Baldomá, S. B., Moner Al Chawa, M., de Benito, C., Stavrinides, S. G., Suñé, J. and Chua, L. O. (2021), 'On the Thermal Models for Resistive Random Access Memory Circuit Simulation', *Nanomaterials* **11**(5), 1261. doi: [10.3390/nano11051261](https://doi.org/10.3390/nano11051261) (page 95).
- Ronneberger, O., Fischer, P. and Brox, T. (2015), U-Net: Convolutional Networks for Biomedical Image Segmentation, in N. Navab, J. Hornegger, W. M. Wells and A. F. Frangi, eds, 'Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015', Lecture Notes in Computer Science, Springer International Publishing, pp. 234–241. doi: [10.1007/978-3-319-24574-4_28](https://doi.org/10.1007/978-3-319-24574-4_28) (page 114).
- Rumsey, S. (2019), Capacity Considerations for Data Storage in Memristor Arrays, PhD thesis, University of Toronto, Toronto, Canada. (page 53).
- Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A. and Chen, X. (2016), 'Improved Techniques for Training GANs', *arXiv:1606.03498 [cs]*. (pages 58, 117).
- Schrödinger, E. (1992), *What Is Life?: With Mind and Matter and Autobiographical Sketches*, Canto, Cambridge University Press, Cambridge. doi: [10.1017/CBO9781139644129](https://doi.org/10.1017/CBO9781139644129) (page 26).
- Shamma, S. A. (1985), 'Speech processing in the auditory system. II: Lateral inhibition and the central processing of speech evoked activity in the auditory nerve', *The Journal of the Acoustical Society of America* **78**(5), 1622–1632. doi: [10.1121/1.392800](https://doi.org/10.1121/1.392800) (page 39).
- Shannon, C. E. (1948), 'A Mathematical Theory of Communication', *Bell System Technical Journal* **27**(3), 379–423. doi: [10.1002/j.1538-7305.1948.tb01338.x](https://doi.org/10.1002/j.1538-7305.1948.tb01338.x) (pages 50, 51).
- Shomorony, I. and Avestimehr, A. S. (2013), 'Worst-Case Additive Noise in Wireless Networks', *IEEE Transactions on Information Theory* **59**(6), 3833–3847. doi: [10.1109/TIT.2013.2248875](https://doi.org/10.1109/TIT.2013.2248875) (page 54).
- Smith, A. and Downey, J. (2019), A communication channel density estimating generative adversarial network, in '2019 IEEE Cognitive Communications for Aerospace Applications Workshop, CCAAW 2019'. doi: [10.1109/CCAAW.2019.8904907](https://doi.org/10.1109/CCAAW.2019.8904907) (page 54).
- Sønderby, C. K., Raiko, T., Maaløe, L., Sønderby, S. K. and Winther, O. (2016), Ladder variational autoencoders, in 'Proceedings of the 30th International Conference on Neural Information Processing Systems', NIPS'16, Curran Associates Inc., pp. 3745–3753. (page 58).
- Song, Y. and Ermon, S. (2020), Improved Techniques for Training Score-Based Generative Models, in 'Advances in Neural Information Processing Systems', Vol. 33, Curran Associates, Inc., pp. 12438–12448. (page 25).

- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R. (2014), ‘Dropout: A Simple Way to Prevent Neural Networks from Overfitting’, *Journal of Machine Learning Research* **15**(56), 1929–1958. (page 59).
- Stathopoulos, S., Michalas, L., Khiat, A., Serb, A. and Prodromakis, T. (2018), A comprehensive technology agnostic RRAM characterisation protocol, Technical Report August, University of Southampton. (page 47).
- Stern, P. (2007), ‘The Winners Recall Memories’, *Science’s STKE* **2007**(383), tw142. doi: [10.1126/stke.3832007tw142](https://doi.org/10.1126/stke.3832007tw142) (page 39).
- Strachan, J. P., Torrezan, A. C., Miao, F., Pickett, M. D., Yang, J. J., Yi, W., Medeiros-Ribeiro, G. and Williams, R. S. (2013), ‘State Dynamics and Modeling of Tantalum Oxide Memristors’, *IEEE Transactions on Electron Devices* **60**(7), 2194–2202. doi: [10.1109/TED.2013.2264476](https://doi.org/10.1109/TED.2013.2264476) (page 86).
- Strukov, D. B., Snider, G. S., Stewart, D. R. and Williams, R. S. (2008), ‘The missing memristor found’, *Nature* **453**(7191), 80–83. doi: [10.1038/nature06932](https://doi.org/10.1038/nature06932) (pages 40, 42, 47, 67, 86).
- Strukov, D. B. and Williams, R. S. (2009), ‘Exponential ionic drift: Fast switching and low volatility of thin-film memristors’, *Applied Physics A* **94**(3), 515–519. doi: [10.1007/s00339-008-4975-3](https://doi.org/10.1007/s00339-008-4975-3) (pages 42, 70, 72).
- Sun, C., Cai, K., Song, G., Quek, T. Q. S. and Fei, Z. (2021), ‘Belief Propagation based Joint Detection and Decoding for Resistive Random Access Memories’, *IEEE Transactions on Communications* p. 34. (page 53).
- Sun, H., Zhu, T., Zhang, Z., Xiong, D. J. P. and Zhou, W. (2021), ‘Adversarial Attacks Against Deep Generative Models on Data: A Survey’, *IEEE Transactions on Knowledge and Data Engineering* . doi: [10.1109/TKDE.2021.3130903](https://doi.org/10.1109/TKDE.2021.3130903) (page 159).
- Theunissen, F. and Miller, J. P. (1995), ‘Temporal encoding in nervous systems: A rigorous definition’, *Journal of Computational Neuroscience* **2**(2), 149–162. doi: [10.1007/BF00961885](https://doi.org/10.1007/BF00961885) (page 37).
- Tishby, N. and Zaslavsky, N. (2015), ‘Deep Learning and the Information Bottleneck Principle’, *arXiv:1503.02406 [cs]* . (page 59).
- Tolstikhin, I., Bousquet, O., Gelly, S. and Schoelkopf, B. (2018), Wasserstein Auto-Encoders, in ‘International Conference on Learning Representations’. (pages 32, 58, 168, 169).
- Tung, T.-Y., Kurka, D. B., Jankowski, M. and Gündüz, D. (2022), DeepJSCC-Q: Channel Input Constrained Deep Joint Source-Channel Coding, in ‘ICC 2022 - IEEE International Conference on Communications’, pp. 3880–3885. doi: [10.1109/ICC45855.2022.9838671](https://doi.org/10.1109/ICC45855.2022.9838671) (page 52).
- van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A. and Kavukcuoglu, K. (2016), ‘WaveNet: A Generative Model for Raw Audio’, *arXiv:1609.03499 [cs]* . (pages 57, 108).

- van den Oord, A., Vinyals, O. and Kavukcuoglu, K. (2017), Neural discrete representation learning, in 'Proceedings of the 31st International Conference on Neural Information Processing Systems', NIPS'17, Curran Associates Inc., pp. 6309–6318. (page 58).
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. and Polosukhin, I. (2017), Attention is all you need, in 'Proceedings of the 31st International Conference on Neural Information Processing Systems', NIPS'17, Curran Associates Inc., pp. 6000–6010. (pages 25, 58, 114).
- von Neumann, J. (1993), 'First draft of a report on the EDVAC', *IEEE Annals of the History of Computing* 15(4), 27–75. doi: 10.1109/85.238389 (pages 40, 45, 62).
- Vongehr, S. and Meng, X. (2015), 'The Missing Memristor has Not been Found', *Scientific Reports* 5(1), 11657. doi: 10.1038/srep11657 (page 40).
- Wang, C., Si, Z., Jiang, X., Malik, A., Pan, Y., Stathopoulos, S., Serb, A., Wang, S., Prodromakis, T. and Papavassiliou, C. (2022), 'Multi-State Memristors and Their Applications: An Overview', *IEEE Journal of Emerging and Selected Topics in Circuits and Systems* . doi: 10.1109/JETCAS.2022.3223295 (page 47).
- Wang, Z., Joshi, S., Savel'ev, S. E., Jiang, H., Midya, R., Lin, P., Hu, M., Ge, N., Strachan, J. P., Li, Z., Wu, Q., Barnell, M., Li, G. L., Xin, H. L., Williams, R. S., Xia, Q. and Yang, J. J. (2017), 'Memristors with diffusive dynamics as synaptic emulators for neuromorphic computing', *Nature Materials* 16(1), 101–108. doi: 10.1038/nmat4756 (pages 74, 75, 100).
- Waser, R. and Aono, M. (2007), 'Nanoionics-based resistive switching memories', *Nature Materials* 6(11), 833–840. doi: 10.1038/nmat2023 (page 42).
- Wei, Z., Takagi, T., Kanzawa, Y., Katoh, Y., Ninomiya, T., Kawai, K., Muraoka, S., Mitani, S., Katayama, K., Fujii, S., Miyanaga, R., Kawashima, Y., Mikawa, T., Shimakawa, K. and Aono, K. (2011), Demonstration of high-density ReRAM ensuring 10-year retention at 85°C based on a newly developed reliability model, in '2011 International Electron Devices Meeting', pp. 31.4.1–31.4.4. doi: 10.1109/IEDM.2011.6131650 (pages 133, 137).
- White, R. H. (1992), 'Competitive hebbian learning: Algorithm and demonstrations', *Neural Networks* 5(2), 261–275. doi: 10.1016/So893-6080(05)80024-3 (pages 39, 157).
- Xu, J., Ai, B., Chen, W., Yang, A., Sun, P. and Rodrigues, M. (2021), 'Wireless Image Transmission Using Deep Source Channel Coding With Attention Modules', *IEEE Transactions on Circuits and Systems for Video Technology* . doi: 10.1109/TCSVT.2021.3082521 (pages 52, 133, 134, 162).
- Xu, J., Tung, T.-Y., Ai, B., Chen, W., Sun, Y. and Gündüz, D. (2023), 'Deep Joint Source-Channel Coding for Semantic Communications', *IEEE Communications Magazine* 61(11), 42–48. doi: 10.1109/MCOM.004.2200819 (pages 52, 162).
- Yang, J. J., Pickett, M. D., Li, X., Ohlberg, D. A. A., Stewart, D. R. and Williams, R. S. (2008), 'Memristive switching mechanism for metal/oxide/metal nanodevices', *Nature Nanotechnology* 3(7), 429–433. doi: 10.1038/nnano.2008.160 (pages 43, 47).

- Yang, Y., Li, Y., Zhang, W., Qin, F., Zhu, P. and Wang, C.-X. (2019), 'Generative-Adversarial-Network-Based Wireless Channel Modeling: Challenges and Opportunities', *IEEE Communications Magazine* **57**(3), 22–27. doi: [10.1109/MCOM.2019.1800635](https://doi.org/10.1109/MCOM.2019.1800635) (page [159](#)).
- Ye, H., Li, G. Y., Juang, B.-H. F. and Sivanesan, K. (2018), Channel Agnostic End-to-End Learning Based Communication Systems with Conditional GAN, in '2018 IEEE Globecom Workshops (GC Wkshps)', pp. 1–5. doi: [10.1109/GLOCOMW.2018.8644250](https://doi.org/10.1109/GLOCOMW.2018.8644250) (pages [54](#), [107](#), [162](#)).
- Ye, H., Liang, L., Li, G. Y. and Juang, B.-H. (2020), 'Deep Learning-Based End-to-End Wireless Communication Systems With Conditional GANs as Unknown Channels', *IEEE Transactions on Wireless Communications* **19**(5), 3133–3143. doi: [10.1109/TWC.2020.2970707](https://doi.org/10.1109/TWC.2020.2970707) (pages [54](#), [107](#), [159](#)).
- Yoon, J. and Jarrett, D. (2019), 'Time-series Generative Adversarial Networks', *Advances in Neural Information Processing Systems* pp. 5508–5518. (page [108](#)).
- Zarcone, R., Paiton, D., Anderson, A., Engel, J., Wong, H. S. and Olshausen, B. (2018), Joint source-channel coding with neural networks for analog data compression and storage, in 'Data Compression Conference Proceedings', Vol. 2018-March, pp. 147–156. doi: [10.1109/DCC.2018.00023](https://doi.org/10.1109/DCC.2018.00023) (pages [53](#), [133](#), [142](#)).
- Zarcone, R. V., Engel, J. H., Burc Eryilmaz, S., Wan, W., Kim, S., BrightSky, M., Lam, C., Lung, H.-L., Olshausen, B. A. and Philip Wong, H.-S. (2020), 'Analog Coding in Emerging Memory Systems', *Scientific Reports* **10**(1), 6831. doi: [10.1038/s41598-020-63723-z](https://doi.org/10.1038/s41598-020-63723-z) (page [53](#)).
- Zha, J., Huang, H. and Liu, Y. (2016), 'A Novel Window Function for Memristor Model With Application in Programming Analog Circuits', *IEEE Transactions on Circuits and Systems II: Express Briefs* **63**(5), 423–427. doi: [10.1109/TCSII.2015.2505959](https://doi.org/10.1109/TCSII.2015.2505959) (page [47](#)).
- Zheng, X., Zarcone, R., Paiton, D., Sohn, J., Wan, W., Olshausen, B. and Wong, H. P. (2018), 'Error-Resilient Analog Image Storage and Compression with Analog-Valued RRAM Arrays : An Adaptive Joint Source-Channel Coding Approach', *2018 IEEE International Electron Devices Meeting (IEDM)* **4**, 71–74. (page [53](#)).
- Zhou, C., Kadambi, P., Mattina, M. and Whatmough, P. N. (2020), 'Noisy Machines: Understanding Noisy Neural Networks and Enhancing Robustness to Analog Hardware Errors Using Distillation', *arXiv:2001.04974 [cs, stat]* . (pages [45](#), [54](#)).
- Zidan, M. A., Fahmy, H. A. H., Hussain, M. M. and Salama, K. N. (2013), 'Memristor-based memory: The sneak paths problem and solutions', *Microelectronics Journal* **44**(2), 176–183. doi: [10.1016/j.mejo.2012.10.001](https://doi.org/10.1016/j.mejo.2012.10.001) (page [44](#)).

MULTIPLE SAMPLE DISCRIMINATION ABLATION RESULTS

We here include results for the same experiments performed in Chapter 4, using a value of $n = 1$ for multiple sample discrimination for both the main discriminator and the delay discriminator. The sub-optimality of these results demonstrates the benefits of the multiple sample discrimination technique, which was introduced in Section 4.3.4. A comparison of the empirically estimated statistics of the conditional distribution induced by the **cGAN** is shown in Figure A.1. A comparison of the empirically sampled final resistance values for a range of delays and input resistances, plotted alongside those derived from the ground truth model, is shown in Figure A.2.

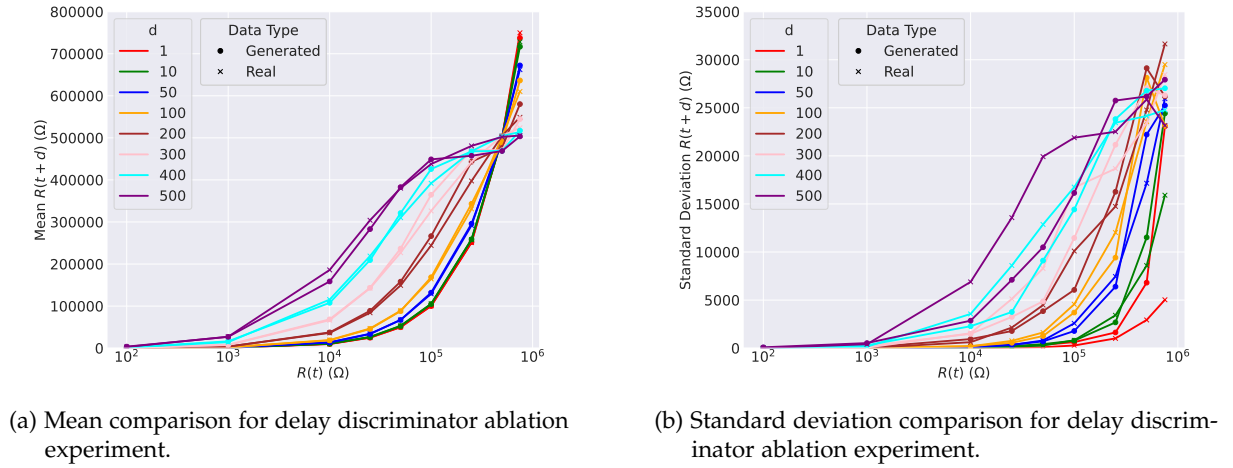
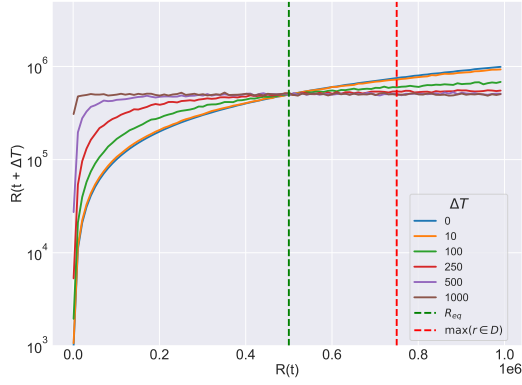
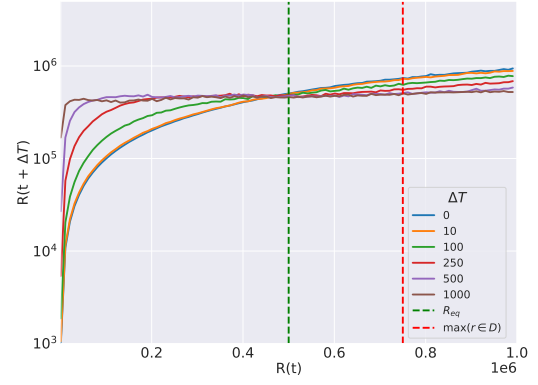


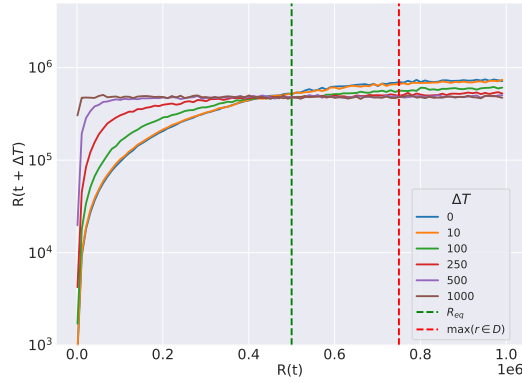
Figure A.1: Means and standard deviations of the final resistance conditioned on different combinations of initial resistances and delays, for the value of $n = 1$ for multiple sample discrimination for both the main discriminator and the delay discriminator. These can be compared to those in Figure 4.9 of Chapter 4, highlighting the improved moment matching when using a value of $n > 1$ for multiple sample discrimination.



(a) Ground truth model.



(b) Multiple sample discrimination ablation experiment.



(c) Multiple sample discrimination ablation experiment with recursive evaluation.

Figure A.2: Empirically sampled final resistances for given initial and final resistance conditions. Figure A.2a shows the statistics for the ground truth event-based model. Figure A.2b shows statistics for the cGAN. Figure A.2c shows statistics for the cGAN model evaluated recursively with a delay of $d/10$ and 10 recursive steps. These results can be compared to those in Figure 4.10 of Chapter 4.