

Imperial College London
Department of Computing

Deep Representation Learning for Audio-Visual Speech Recognition and Synthesis

Andreas Zinonos

Supervised by Prof. Maja Pantic and Dr. Stavros Petridis

Submitted in part fulfilment of the requirements for the degree of
Doctor of Philosophy in Computing and the Diploma of Imperial College

December 2025

Declaration of originality

I confirm that, to the best of my knowledge, this thesis and the research presented within it are the result of my own independent work. Where the work of others has been drawn upon, it has been clearly acknowledged through proper citation. This thesis has not been submitted for any other purposes.

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 license, 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 license terms clear to others by naming the license and linking to the license 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 license or permitted under UK Copyright Law.

Abstract

Deep learning has grown rapidly in capability and impact over the past decade, now achieving industry-disrupting results that rival or surpass traditional methods across many domains. While most early progress relied heavily on large labelled datasets, recent advances show that powerful models can be trained by exploiting the abundance of unlabelled data available in the wild. Audio-visual recordings of people speaking are particularly well suited for this paradigm, as they provide rich and naturally aligned cross-modal signals. This thesis investigates how large-scale unlabelled audio-visual data of talking humans can be used to improve two important tasks: speech recognition and lip synchronization. We first extend the self-supervised learning framework RAVeN to multilingual settings, studying how cross-lingual audio-visual pre-training benefits visual speech recognition, especially for languages with limited labelled resources. We then propose BRAVeN, an improved version of RAVeN that more effectively exploits the complementary structure of audio and video through architectural refinements and modality-aware masking, yielding substantial gains in both visual and audio speech recognition, particularly under low-label conditions. Finally, we turn to generative modeling and introduce FlashLips, a two-stage lip synchronization system that produces temporally accurate lip motion at over 100 FPS, while matching the quality of significantly slower iterative diffusion-based approaches. Together, these contributions demonstrate that unlabelled audio-visual data can be leveraged effectively for both discriminative and generative tasks, enabling stronger speech recognition systems and faster, high-fidelity lip-sync generation without the need for expensive manual annotation.

Acknowledgements

Completing this PhD has been a long and often challenging journey – far more challenging than I ever anticipated. Along the way, I learned a great deal, gained experiences I could never have imagined, and met many wonderful people who each contributed to this process in their own way. I am deeply grateful to all those who supported me throughout these four years, as well as to those who were part of my life long before it began.

First and foremost, I would like to thank my supervisor, Maja, for giving me the opportunity to pursue a PhD and a research position at Imperial College London and Meta, in what is arguably the most rapidly evolving field of the decade. I am especially grateful to Stavros, whose mentorship, insight, and consistent involvement shaped this thesis in fundamental ways. His dedication, support, and dependability were invaluable, and I could always count on him for guidance.

I would also like to thank my friends and colleagues at Imperial and Meta, who made this experience far more manageable and enjoyable. Special thanks to my main co-author Alex H., whose prior work, expertise, and willingness to help with my endless questions made my path significantly easier. I am also grateful to Rodrigo, the first welcoming face I met at Imperial, who taught me the practical tools and shortcuts that made daily research much smoother. Thank you to Pingchuan for his deep knowledge in speech recognition and for always being open to discussions, and to Nikita for the opportunity to collaborate on a new research direction and for providing guidance throughout that work.

My gratitude also goes to Minsu for offering valuable research direction and for helping me run many experiments along with Jasmina. Thank you to Dino, Adriana, and Honglie, whose ideas, discussions, and time consistently pushed my work forward. I am also thankful to Antoni for his talent in finding clear solutions to not-so-clear problems. I would like to thank the Meta band – Viktor, Zoe, and Rodrigo – for the many hours of jamming, even if we never quite made it to the stage. Thank you as well to Michal for his help with diffusion models and our ongoing table tennis rivalry. I am grateful to Lloyd and the CSG department, and to Amani and Teresa, who were always quick to assist with every issue or request at Imperial. A special mention goes to my friend Patrick, who may not have contributed much directly to this PhD, but was effectively my ChatGPT (before it existed) during my master's, teaching

me the fundamentals of machine learning and ultimately setting me on the path toward this doctorate – I still owe him beers for life. Thank you also to Jie, Alex S., Leo, Marija, Ruslan, and Kacper for their support at various points throughout this journey.

Finally, I want to thank all my friends who provided much-needed distraction from PhD life: Andreas G., Andreas E., Antonis, Nikolas, Michalis, Renos, Marios, Christos, Lambros, Valentin, Jean, and Steven. I am also thankful to my siblings, Angelos and Despoina, and to my grandparents, for always being willing to help in their own ways. And of course, my deepest gratitude goes to my parents, who have supported me throughout this journey and my entire life with unconditional love, encouragement, and a home to return to whenever I needed it. For that, I will be forever grateful.

To my family, and my dog Daisy.

CONTENTS

Abstract	iv
Acknowledgements	v
List of Tables	xvii
List of Figures	xxiv
1 Introduction	1
1.1 Motivation	1
1.2 Contributions	5
1.3 Publications	7
2 Background	8
2.1 Datasets	9
2.1.1 Speech Recognition	10
2.1.2 Lip Synchronization	11

2.1.3	Summary of Datasets	14
2.2	Data Pre-processing	15
2.2.1	Audio Pre-Processing	15
2.2.2	Video Pre-Processing	15
2.2.3	Text Pre-Processing	18
2.3	Neural Architectures	18
2.3.1	Fully Connected Neural Networks	18
2.3.2	Convolutional Neural Networks	20
2.3.3	ResNets	25
2.3.4	U-Nets	27
2.3.5	Transformers	28
2.4	Representation Learning	31
2.4.1	Supervised Learning	32
2.4.2	Unsupervised Learning	32
2.4.3	Self-supervised Learning	34
2.5	Generative Modeling	36
2.5.1	Autoencoders	36
2.5.2	Generative Adversarial Networks	39
2.5.3	Diffusion Models	42
2.5.4	Flow Matching Models	46
2.6	Optimisers	50
2.6.1	Stochastic Gradient Descent (SGD)	50

2.6.2	Adam	50
2.6.3	AdamW	51
2.6.4	Learning Rate Schedules	51
2.7	Loss Functions	52
2.7.1	Speech Recognition	52
2.7.2	Lip Synchronization	55
2.8	Evaluation Metrics	57
2.8.1	Speech Recognition	57
2.8.2	Lip Synchronization	58
3	Learning Cross-lingual Visual Speech Representations	62
3.1	Introduction	63
3.2	Datasets	64
3.3	Method	65
3.3.1	Pre-training	65
3.3.2	Fine-tuning	66
3.4	Experiments	67
3.4.1	Pre-processing	67
3.4.2	Data augmentation	67
3.4.3	Masking	67
3.4.4	Training settings	67
3.4.5	Fine-tuning	68
3.5	Results	68

3.5.1	Multilingual pre-training surpasses monolingual	68
3.5.2	Multilingual pre-training surpasses English-only pre-training with the same number of hours	69
3.5.3	Monolingual pre-training surpasses multilingual with same hours	69
3.5.4	Related language pre-training performs better	71
3.5.5	Representations on unseen languages	71
3.6	Limitations	72
3.7	Conclusion	72
4	BRAVE_n: Improving Self-Supervised Pre-Training for Speech Recognition	73
4.1	Introduction	74
4.2	Method	76
4.2.1	Background: RAVE _n	76
4.2.2	BRAVE _n	77
4.2.3	Fine-tuning	78
4.3	Experimental Setup	79
4.3.1	Datasets	79
4.3.2	Architecture	79
4.3.3	Data pre-processing	80
4.3.4	Training details	80
4.4	Results	83
4.4.1	Low-resource setting	83
4.4.2	High-resource setting	83

4.4.3	Audio-visual results	84
4.4.4	Ablations	85
4.5	Limitations	85
4.6	Conclusion	86
5	FlashLips: 100-FPS Mask-Free Latent Lip-Sync using Reconstruction	87
5.1	Introduction	88
5.2	Related Work	92
5.2.1	Audio-driven Portrait Animation	92
5.2.2	Audio-driven Lip Synchronization	93
5.3	Method	94
5.3.1	Stage 1: Latent Visual Editor	94
5.3.2	Stage 2: Audio-to-Lips with Flow Matching	98
5.3.3	Architecture Details	99
5.4	Experiments	101
5.4.1	Datasets	101
5.4.2	Training Details	102
5.4.3	Evaluation Metrics	104
5.4.4	Quantitative Evaluation	105
5.4.5	Qualitative Evaluation	106
5.4.6	User Study	107
5.4.7	Mask Removal: Quantitative Impact	107
5.5	Ablation Study	109

5.6	Limitations	110
5.7	Conclusion	111
6	Conclusion	116
6.1	Summary	116
6.2	Ethical Considerations	118
6.2.1	Speech Recognition	118
6.2.2	Lip Synchronization	119
6.3	Future Work	120

LIST OF TABLES

2.1	Summary of audio-visual datasets used in our projects. Datasets are grouped by whether they were used for speech recognition or lip synchronization. For multilingual datasets, only the relevant language subsets used are listed. *Used both with and without labels depending on the pipeline stage.	14
3.1	Character Error Rate (CER) of visual-only models for different languages. We pre-train models on either one language or six, and fine-tune on each language independently. English-only refers to the English monolingual model.	69
3.2	CER on low-resource languages. We pre-train six monolingual models with 30 hours of data randomly sampled from each language, and one multilingual model with the same hours of pre-training data by sampling 5 hours from each language randomly. We compare the results in the case where the languages are low-resource.	70
3.3	CER comparison when fine-tuning Spanish, pre-training on Spanish with added languages. We pre-train models on three different pairs of languages without the use of any labelled data, and then fine-tune the visual model using a total of 62 hours of labelled video clips in Spanish.	70

3.4	CER on unseen languages. We pre-train two multilingual models: one excluding Portuguese and one excluding Spanish, using the same number of pre-training hours as the Multilingual RE model we are comparing to.	71
4.1	Configuration of the transformer encoders.	79
4.2	LRS3 results: low-resource. “Unlab hrs”/“lab hrs” denote unlabelled hours/labelled hours. Self-supervised methods use labelled data for pre-training (without labels) along with unlabelled data (if any). “LM” denotes language model. “ST” denotes self-training. [†] Uses an additional 3,846/452 hours of audio/audio-text data. [‡] Uses the (smaller) Base model.	81
4.3	LRS3 results: high-resource. “Unlab hrs”/“lab hrs” denote unlabelled hours/labelled hours. Self-supervised methods use labelled data for pre-training (without labels) along with unlabelled data (if any). “LM” denotes language model. “ST” denotes self-training. [*] Includes synthetic visual data generated using labelled audio. [†] Uses an additional 3,846/452 hours of audio/audio-text data. [‡] Uses the (smaller) Base model.	82
4.4	Audio-visual results. We show low-resource results (WER %) (Base models) with different signal-to-noise ratio (SNR).	84
4.5	Ablations. We show results for the low-resource setting using the Base model. .	85
5.1	Architecture of the U-Net base model. Each ResNet2D block consists of GroupNorm (GN, 32 groups), SiLU activation, and two Conv2d layers ($k = 3, s = 1, p = 1$).	99
5.2	Architecture of the Transformer base model. MHSA stands for Multi-Head Self-Attention.	100
5.3	Architecture of the Flow Matching Transformer (FMT). d_p is the lip-pose latent dimension, d_h is the hidden size. MHSA stands for Multi-Head Self-Attention.	100

5.4	Quantitative comparison. Comparison on reconstruction and cross-audio scenarios over 100 randomly sampled reconstruction videos and 100 cross-audio pairs from HDTF, CelebV-HQ, and CelebV-Text. Best results are bold ; second-best are <u>underlined</u>	101
5.5	Quantitative comparison on VBench. Video quality evaluation using VBench Huang <i>et al.</i> , 2023 metrics on 100 randomly sampled reconstruction videos and 100 cross-audio pairs from HDTF, CelebV-HQ and CelebV-Text. Metrics defined in Section 5.4.3.	102
5.6	Inference speed. Speed comparison in frames per second (FPS). “Speedup” denotes the inference speed gain of our fastest model (FlashLips – U-Net) over each method. Measured on the same clip: 5 warm-ups, then 10 runs to average FPS.	103
5.7	Full ablation study. Ablation study of our mask and mask-free models, and different numbers of references for the audio-to-latent model for reconstruction and cross-audio. Metrics computed on 100 randomly sampled reconstruction videos and 100 cross-audio pairs from HDTF, CelebV-HQ and CelebV-Text. . .	104
5.8	Lips encoder ablation. V1 (frozen expression encoder) saturates near 8D; adding a lips-crop residual (V2) improves reconstruction but reduces cross-audio ID. The 12D setting (V1 8D + V2 4D) offers the best quality–disentanglement trade-off.	105

LIST OF FIGURES

2.1	An example of a waveform. Image adapted from (PyTorch Team, 2024) . . .	15
2.2	Video pre-processing pipeline for speech recognition. Overview of the stages applied to each input video.	16
2.3	Examples of speech recognition video pre-processing. Example frames from two different videos before and after processing.	16
2.4	Video pre-processing pipeline for lip synchronization. Overview of the stages applied to each input video.	17
2.5	Examples of lip synchronization video pre-processing. Example frames from three different videos after processing.	17
2.6	A simple Fully Connected Neural Network (FCNN). It consists of two hidden layers and an output layer with a single output neuron. Each neuron is connected to all neurons in the previous and next layer. Figure from (Karpathy <i>et al.</i> , 2016).	19
2.7	Example of a 2D convolution. A padding of 1 is applied to the input, shown as yellow cells with value 0. Each 3×3 region of the padded input is multiplied element-wise with the 3×3 kernel, and the results are summed to produce a single output value. Figure adapted from (Godoy, 2021).	21

2.8	Example of max pooling. The figure shows a 2×2 max pooling with stride 2, where only the maximum value in each window is kept. Figure from (Godoy, 2021).	23
2.9	Illustration of a transposed convolution. The figure shows up-sampling of a 2×2 input to a 4×4 output using a 3×3 kernel with stride 1. Zeros are inserted to expand the input, and then convolved. Figure from (Dumoulin and Visin, 2018).	24
2.10	Example of a basic residual block. The block consists of two layers, ReLU activations and a skip connection. Figure from (He <i>et al.</i> , 2016).	25
2.11	Architecture of the ResNet-34. The figure shows all of its 34 layers and skip connections. Figure adapted from (He <i>et al.</i> , 2016).	26
2.12	Illustration of the U-Net architecture. It consists of an encoder-decoder structure with skip connections. Downsampling is performed via strided convolution or pooling; upsampling uses transposed convolutions or interpolation. Figure from (Ronneberger <i>et al.</i> , 2015).	27
2.13	Illustration of the transformer architecture. The figure shows the encoder-decoder structure, with their various components. Figure from (Vaswani <i>et al.</i> , 2017).	29
2.14	Single and multi-head attention mechanisms. Multi-head attention allows the model to capture diverse contextual relationships by projecting the input into different representation subspaces. Figure from (Vaswani <i>et al.</i> , 2017).	30
2.15	A high-level comparison of unsupervised and supervised learning. Unsupervised learning groups similar data points into clusters without using labels, while supervised learning uses labels to find boundaries for classification. Figure from (Lawtomated, 2021).	33

2.16	General pipeline of self-supervised learning. Embeddings are learned by training deep neural networks on a pretext task using pseudo-labels. Once training is complete, the resulting pre-trained model is used to extract embeddings, which are then transferred to downstream tasks that rely on real labels. Figure from (Jing and Tian, 2020).	34
2.17	Diagram of a vanilla autoencoder. The autoencoder takes an input $\mathbf{x}$ into the encoder g_ϕ to produce latent $\mathbf{z}$, and then gives it to decoder f_θ to get the reconstruction $\mathbf{x}'$. Figure from (Weng, 2018).	37
2.18	Diagram of a variational autoencoder. The diagram shows the VAE architecture with the encoder outputting $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$, sampling $\mathbf{z}$ via reparameterisation trick, and reconstruction $\mathbf{x}'$ via decoder. Figure from Weng, 2018.	38
2.19	Diagram of a GAN. The framework shows a Generator producing synthetic images from random noise and a Discriminator trying to distinguish between the real and synthetic samples. Figure from (Buff ML, 2022).	39
2.20	A collection of generated images from a GAN. The network was trained on the ImageNet-1K dataset (Deng <i>et al.</i> , 2009).	40
2.21	Diagram of a conditional GAN. The generator is conditioned by extracted audio and identity image features, producing facial motion frames. Figure from (Vougioukas <i>et al.</i> , 2019).	41
2.22	A figure illustrating the forward and reverse diffusion process. The input $\mathbf{x}_0$ is progressively corrupted and denoised back to its original form (Song <i>et al.</i> , 2021).	42
2.23	Some generated samples from DDPMs. Samples are from the CelebA-HQ (Karras <i>et al.</i> , 2018) (left) and CIFAR10 (Krizhevsky, 2009) (right) datasets. Figure from (Ho <i>et al.</i> , 2020b).	43
2.24	Pseudocode for training and sampling in DDPMs. Algorithms from (Ho <i>et al.</i> , 2020b).	45

2.25	A figure comparing trajectories of Continuous Normalizing Flows (CNFs). (Left) CNF trajectories trained with three different objectives on 2D checkerboards. Optimal Transport (OT) based flows recover structure earlier, while diffusion-based objectives converge more slowly. (Right) Flow matching with OT produces higher-quality samples with fewer sampling steps. Figure from (Lipman <i>et al.</i> , 2022).	46
2.26	Pseudocode for sampling in flow matching.	48
2.27	Some generated samples with flow matching with optimal transport. Samples are of resolution 128×128 and generated unconditionally from ImageNet (Deng <i>et al.</i> , 2009). Figure from (Lipman <i>et al.</i> , 2022).	49
3.1	Pre-training and fine-tuning pipelines. We use RAVeN (Haliassos <i>et al.</i> , 2023) to pre-train models using audio and video from multiple languages. Stop-gradient and Exponential Moving Average are denoted by <i>sg</i> and <i>EMA</i> , respectively. We then initialise the visual-only encoder using the weights from the pre-training stage and fine-tune the entire network using the language-specific dataset.	65
4.1	BRAVeN overview. BRAVeN uses targets that are the average of the outputs of the Transformer encoder blocks of the teacher networks. The video student predicts the audio targets via a 1-block predictor. In contrast, the audio student uses 2-block predictors and uses both a cross- as well as a within-modal loss, whose weight is twice as large as that of the cross-modal loss. The input masking for audio is stronger than the masking applied for video.	76
5.1	FlashLips results. Selected results of source and driver pairs, generated using our transformer-based model.	89
5.2	Visualization of quantitative evaluation. Comparison of eight different lip-sync models in the cross-audio setting on seven key metrics. All results are normalized, with the best-performing model scaled to the outer edge, and the worst scaled towards the center.	90

5.3	Overview of FlashLips. Stage 1 trains a one-step latent-space editor: first via masked reconstruction, then via a mask-free self-refinement step that learns to localize edits without segmentation. Stage 2 trains an audio-to-lips model that predicts the lips-pose vector used in Stage 1. At inference, predicted lip poses drive the LipsChange network to produce lip-synced frames in a single pass.	94
5.4	Lips encoder. A frozen expression encoder with an MLP projector and a mouth-crop CNN produce an 8D+4D lips vector. A distilled ResNet-34 replicates this mapping on inference.	95
5.5	Human preference evaluation. We conducted a user study comparing FlashLips against randomly selected baseline models. Participants indicated their preference across two criteria: Visual Quality and Lip Sync. The chart displays the number of responses favoring our model (Ours), the competing model (Other), or neither (Same).	108
5.6	Limitations. Examples illustrating typical failure cases under challenging conditions, including generating facial hair and teeth details, occlusions, and artifacts caused by the SDXL VAE.	111
5.7	Qualitative comparison – cross-audio. Comparison with other lip-sync methods for cross-audio. The first row shows the source video, the second row shows the video corresponding to the audio driver. The following rows display the inferred lip-synced videos by each method.	112
5.8	Qualitative comparison – reconstruction. Comparisons with other lip-sync methods for reconstruction. The first row shows the source video; the following rows display the inferred lip-synced videos by each method.	113
5.9	Lip-sync results across varying facial pose combinations. Each triplet shows a source video, video corresponding to the audio driver, and the resulting prediction.	114

5.10 **Lip-sync results on out-of-distribution (OOD) faces.** The top block of images shows results on generated human faces, while the lower block shows results on non-human or stylized faces. Our method maintains consistent lip synchronization and natural articulation across both domains. 115

CHAPTER 1

INTRODUCTION

Contents

1.1	Motivation	1
1.2	Contributions	5
1.3	Publications	7

1.1 Motivation

Human communication is an inherently multimodal process. Speech is rarely conveyed through sound alone; it is also accompanied by facial expressions, gestures, and other visual signals that jointly shape how messages are transmitted and understood. For instance, observing lip movements can disambiguate words in noisy environments, while facial expressions provide emotional and contextual cues that modulate the meaning of spoken words. The combination of audio and visual signals plays a central role in human communication, enhancing both comprehension and expressiveness. As such, machines designed to interpret human behaviour

must be equipped not only to process sound and video in isolation, but to learn the shared structure that binds them together.

The field of audio-visual learning seeks to enable this understanding by developing models that learn from synchronised audio and visual inputs — typically videos of people speaking. This data contains rich, naturally aligned structure across modalities: utterances correspond to mouth movements, voice animation aligns with head motion, identity and conversational cues span both audio and visuals. Exploiting this structure is essential to tasks such as speech recognition (Ma *et al.*, 2021c; Afouras *et al.*, 2018a) or speaker identification (Nagrani *et al.*, 2017). However, these applications often remain limited by their reliance on supervised learning, which demands expensive labels and often fails to generalise across domains or conditions.

In response, the machine learning community has increasingly turned to *representation learning* – the process of automatically extracting compact, informative, task-general features from raw data (Bengio *et al.*, 2014) while minimising less useful features that would be present in the raw input. The advent of deep learning has significantly advanced representation learning by enabling deep neural networks to extract rich, high-dimensional features from raw data (Goodfellow *et al.*, 2016). Foundational architectures such as Convolutional Neural Networks (CNNs) (LeCun, Bengio, *et al.*, 1995), Recurrent Neural Networks (RNNs) (Hochreiter and Schmidhuber, 1997), and Transformers (Vaswani *et al.*, 2017) have largely eliminated the need for manual feature engineering by learning hierarchical representations directly from raw data. More recent models, including Graph Neural Networks (GNNs) (Kipf and Welling, 2017) and Vision Transformers (ViTs) (Dosovitskiy *et al.*, 2021), continue this trend by extending representation learning to more complex data structures and modalities.

Self-supervised learning has emerged as a powerful strategy for learning from large-scale unlabelled data, by designing pretext tasks that leverage natural structure in the input itself (Balestrieri *et al.*, 2023a; Oord *et al.*, 2018; He *et al.*, 2020; Baevski *et al.*, 2020b). These methods aim to learn useful feature representations without relying on manually annotated labels, making them particularly well-suited to audio-visual domains where labeled data is scarce but unlabelled video is abundant. A broad family of techniques has been developed,

including contrastive methods such as SimCLR (Chen *et al.*, 2020) and MoCo (He *et al.*, 2020), which train models to distinguish between positive (e.g., temporally aligned) and negative (e.g., misaligned) pairs. Other approaches like BYOL (Grill *et al.*, 2020) and DINO (Caron *et al.*, 2021) avoid the need for negative pairs altogether, using momentum encoders or knowledge distillation objectives to align representations across augmented views.

In the audio-visual domain, these ideas have been extended to leverage the natural synchrony between modalities. Models may be trained to align audio and visual streams through cross-modal contrastive learning (Arandjelovic and Zisserman, 2017), to reconstruct masked inputs from context using transformer-based architectures (He *et al.*, 2022; Hsu *et al.*, 2021), or to predict modality-specific representations from one another. For instance, AV-HuBERT (Shi *et al.*, 2022) extends HuBERT’s masked prediction framework to multi-modal speech recognition, combining both acoustic and visual inputs in a unified transformer model. As a result, self-supervised pretraining has become a foundational component in the development of robust, label-efficient systems for speech, vision, and behaviour modeling.

Learning in representation space also serves as a foundation for generative models for audio-visual synthesis (Zhou *et al.*, 2019; Prajwal *et al.*, 2020a; Bigata *et al.*, 2025a,b). *Generative learning* aims to synthesise new content – for example, animating a face to match a given speech segment. As such, talking face videos are a natural application for learning without labels. The data is abundant, naturally synchronised, and richly expressive – capturing both the linguistic content and non-verbal signals of human behaviour. Unlike conventional datasets that rely on labelled text or annotations, these videos can be used to train models directly through unsupervised objectives, enabling the discovery of abstract representations that support the desired target task. Generative models have found applications in entertainment, telepresence, and accessibility, from realistic dubbing of video content to virtual avatars and speech-driven animation (Zhou *et al.*, 2020; Wang *et al.*, 2020). However, many generative models have historically suffered from unstable training, limited controllability, or poor temporal coherence, motivating the exploration of alternative frameworks such as flow- or diffusion-based models (Lipman *et al.*, 2022; Ho *et al.*, 2020a). These approaches offer new ways to model complex dynamics in audio-visual data, potentially enabling fine-grained control over both content and

style of generated behaviour.

Despite advances in both fields, key challenges remain. In self-supervised learning for speech recognition, architectures are predominantly trained and evaluated in monolingual – often English – settings, making it unclear how well speech representations generalise across languages or how multilingual data can be leveraged effectively in low-resource scenarios. Furthermore, existing approaches often under-perform in low-resource regimes, and audio-visual frameworks do not always fully exploit the differing information content of each modality, instead treating them symmetrically. These limitations highlight the need for more data-efficient and robust self-supervised methods that learn transferable representations from raw multi-modal data with minimal reliance on labelled supervision. In generative learning, cross-modal alignment, temporal consistency, identity preservation, and generalisation to unseen speakers or languages remain challenging. While diffusion-based models achieve high visual fidelity, their reliance on iterative sampling results in slow inference, limiting their suitability for real-time applications and motivating the need for faster, more efficient generative frameworks.

This thesis is motivated by the opportunity to advance the aforementioned methods. By developing self-supervised and generative learning frameworks tailored to audio-visual human behaviour, our aim is to reduce reliance on labelled data, improve speech recognition generalisation, and enable fast, high-quality lip synchronization. This will unlock more reliable, efficient, and flexible machine understanding of the behaviours that define human interaction.

1.2 Contributions

We list our contributions to the field below.

1. Learning Cross-lingual Visual Speech Representations

In Chapter 3, we investigate the extent to which visual speech representations learned in a self-supervised audio-visual setting generalise across languages (Zinonos *et al.*, 2023). Building on the RAVEn (Haliassos *et al.*, 2023) framework, we perform controlled multilingual pre-training and fine-tuning experiments using a range of audio-visual datasets containing different languages. Our work provides several interesting insights into cross-lingual transfer and the interplay between language similarity, data quantity, and monolingual vs multilingual pre-training. By systematically varying the amount of training data and the combination of source and target languages, we uncover trends that inform best practices for multilingual and low-resource model development. These findings highlight when and how cross-lingual transfer can be leveraged effectively for practical VSR systems.

2. BRAVEN: Improving Self-supervised Pre-training for Visual and Auditory Speech Recognition

In Chapter 4, we present BRAVEN (Haliassos *et al.*, 2024b), a set of architectural and objective-level improvements to the RAVEn framework for audio-visual self-supervised learning. We introduce the use of average-pool targets from the predictor instead of final-layer features to produce smoother targets, shallower visual predictors to encourage the visual encoder to capture more information in this lower-signal modality, lower visual loss weighting relative to audio to reflect the higher information content of the audio stream, and stronger asymmetric masking in the audio branch to improve robustness. These modifications result in state-of-the-art self-supervised performance on LRS3 using only 30 hours of labelled data, in contrast to other methods that use orders of magnitude more – representing significant improvements over prior work in low-resource settings.

3. FlashLips: 100-FPS Mask-Free Latent Lip-Sync using Reconstruction Instead of Diffusion or GANs

In Chapter 5, we tackle a different task: efficient audio-driven lip synchronization using only unlabeled data. We introduce FlashLips (Zinonos *et al.*, 2025), a two-stage lip synchronization framework that challenges the need for adversarial or diffusion-based systems. In the first stage, we design a lightweight one-step latent editor that performs mouth edits directly in VAE latent space using only reconstruction losses, avoiding the instability of GANs and the sequential cost of diffusion. A self-refinement procedure further improves lip edits by removing reliance on masking. In the second stage, we map audio to a disentangled lips-pose control space via a flow-matching transformer conditioned on extracted audio features, ensuring that audio provides pose information only while appearance is preserved by the visual editor. FlashLips runs at over 100 FPS on a single H100 GPU while matching the visual quality of substantially heavier iterative models, demonstrating that deterministic, one-step generation is both feasible and highly effective for lip-sync.

1.3 Publications

The publications produced during this PhD are listed below in chronological order:

1. **Learning Cross-lingual Visual Speech Representations**

A. Zinonos, A. Haliassos, P. Ma, S. Petridis, M. Pantic

ICASSP 2023

2. **BRAVEN: Improving Self-supervised Pre-training for Visual and Auditory Speech Recognition**

A. Haliassos*, **A. Zinonos***, R. Mira, S. Petridis, M. Pantic

ICASSP 2024

3. **FlashLips: 100-FPS Mask-Free Latent Lip-Sync using Reconstruction Instead of Diffusion or GANs**

A. Zinonos*, M. Stypulkowski, A. Bigata Casademunt, S. Petridis, M. Pantic, N. Drobyshev*

CVPR 2026

* *denotes equal contribution.*

CHAPTER 2

BACKGROUND

Contents

2.1	Datasets	9
2.1.1	Speech Recognition	10
2.1.2	Lip Synchronization	11
2.1.3	Summary of Datasets	14
2.2	Data Pre-processing	15
2.2.1	Audio Pre-Processing	15
2.2.2	Video Pre-Processing	15
2.2.3	Text Pre-Processing	18
2.3	Neural Architectures	18
2.3.1	Fully Connected Neural Networks	18
2.3.2	Convolutional Neural Networks	20
2.3.3	ResNets	25
2.3.4	U-Nets	27

2.3.5	Transformers	28
2.4	Representation Learning	31
2.4.1	Supervised Learning	32
2.4.2	Unsupervised Learning	32
2.4.3	Self-supervised Learning	34
2.5	Generative Modeling	36
2.5.1	Autoencoders	36
2.5.2	Generative Adversarial Networks	39
2.5.3	Diffusion Models	42
2.5.4	Flow Matching Models	46
2.6	Optimisers	50
2.6.1	Stochastic Gradient Descent (SGD)	50
2.6.2	Adam	50
2.6.3	AdamW	51
2.6.4	Learning Rate Schedules	51
2.7	Loss Functions	52
2.7.1	Speech Recognition	52
2.7.2	Lip Synchronization	55
2.8	Evaluation Metrics	57
2.8.1	Speech Recognition	57
2.8.2	Lip Synchronization	58

2.1 Datasets

In this section, we describe the audio-visual datasets used in our work. We divide them into speech recognition datasets and lip synchronization datasets, based on the task for which they

were used. For many of these datasets, we do not use the provided labels and instead use them as unlabelled data depending on the task and pipeline stage. A summary of all datasets, including whether labels were used, is provided in Table 2.1 at the end of the section.

2.1.1 Speech Recognition

AVSpeech

AVSpeech (Ephrat *et al.*, 2018) is a large-scale audio-visual dataset automatically collected from over 290,000 YouTube videos. It contains approximately 4,700 hours of content, spanning over 150,000 speakers across more than 11 languages. The clips range from 3 to 10 seconds in length and are sourced from public lectures, interviews, and instructional content. Each clip contains only a single visible person speaking, for both video and audio. We select language-specific subsets and calculate the number of hours for each using the VoxLingua107 language classifier (Valk and Alumäe, 2021).

VoxCeleb2

VoxCeleb2 (Chung *et al.*, 2018) consists of over 1.1 million utterances from 6,112 speakers, totalling 2,442 hours of audio-visual speech. The dataset was automatically assembled by retrieving interview-style YouTube videos featuring celebrities and public figures. Face verification and speech activity detection are applied to ensure that the face and voice correspond to the same person. The recordings are highly variable in terms of pose, background noise, illumination, and resolution, providing a valuable source of in-the-wild speech. We use an English-only subset filtered using a language detector by (Shi *et al.*, 2022), resulting in 1,323 hours of usable content.

LRS3

LRS3 (Afouras *et al.*, 2018b) is a large multi-modal dataset for visual and audio-visual speech recognition. It uses TED and TEDx talks from YouTube as its source material. It contains more

than 150,000 utterances with high diversity in speaker appearance and acoustic environments, totaling 438 hours, split into 408 hours in the pre-training set, 30 hours in the training-validation set and 0.9 hours in the test set. Videos are processed using a similar pipeline as LRS2 (Chung *et al.*, 2017) but are generally more varied in quality, making it a more challenging dataset.

CMLR

The Chinese Mandarin Lip Reading (CMLR) dataset (Zhao *et al.*, 2019) consists of 102,112 sentence-level clips extracted from news broadcasts on Chinese national television. The training, validation, and test splits contain 60.6, 8.6, and 17.3 hours of content, respectively. Each utterance includes aligned text transcriptions. It is one of the largest publicly available datasets in Mandarin.

Multilingual TEDx

Multilingual TEDx (Salesky *et al.*, 2021) is a large-scale dataset of TEDx talks in eight languages with manually curated transcriptions, with a total duration of 765 hours. The dataset includes language-specific subsets comprising 189 hours of Spanish, 189 hours of French, 164 hours of Portuguese, and 107 hours of Italian. For our purposes, after manually excluding videos where the targeted speaker is not visible, we utilize 62 hours of Spanish, 72 hours of French, 73 hours of Portuguese, and 40 hours of Italian.

2.1.2 Lip Synchronization

HDTF

The High-Definition Talking Face (HDTF) dataset (Zhang *et al.*, 2021) is an in-the-wild audio-visual corpus collected from YouTube videos. It consists of approximately 15.8 hours of 720p–1080p video from around 362 clips and more than 300 speakers, covering roughly 10,000 distinct sentences. Faces are detected, cropped, and resized to 512×512 resolution, resulting in

high-quality talking-head videos with natural variation in identity, head pose, expression, and background.

FEED

FEED (Drobyshev *et al.*, 2024) (Facial Extreme Emotions Dataset) is a recent multi-view video dataset designed to capture a wide spectrum of extreme facial expressions. It contains approximately 6.5 hours of video, with 520 videos from 23 subjects, recorded simultaneously by three cameras at 4K resolution, and includes tasks such as blinks, eye movements, head rotations, tongue motions, and both mild and strong versions of basic facial expressions, as well as more free-form, asymmetric expressions.

NeRSemble

NeRSemble (Kirschstein *et al.*, 2023) is a multi-view video dataset of human heads captured with a custom rig of 16 synchronized machine-vision cameras at 7.1 MP resolution and 73 fps. It comprises more than 4,700 sequences and 220 participants, amounting to roughly 7.5 hours, covering a variety of facial expressions, emotions, head motions, and spoken sentences in a studio setting.

CelebV-Text

CelebV-Text (Yu *et al.*, 2023) is a large-scale facial text–video dataset aimed at text-driven face video generation. It contains 70,000 in-the-wild face-centric clips of around 279 hours, harvested from YouTube, each paired with 20 textual descriptions. The videos cover a broad range of identities, poses, lighting conditions, and motion patterns.

CelebV-HQ

CelebV-HQ (Zhu *et al.*, 2022) is a high-quality celebrity video dataset curated for facial attribute analysis and video generation. It consists of around 65 hours, consisting of 35,666 clips with a minimum resolution of 512×512 , involving 15,653 identities collected from YouTube.

RAVDESS

RAVDESS (Livingstone and Russo, 2018) (Ryerson Audio-Visual Database of Emotional Speech and Song) is a multimodal dataset of acted speech and song. It contains 24 professional actors (12 male, 12 female) speaking in North American English. The dataset includes recordings across multiple emotions and intensity levels, resulting in a total of 7,356 files across modalities: audio-only, video-only, and audio-visual — of which we only use the audio-visual portion. Video files are provided in 720p format, audio in 48 kHz WAV.

Hallo3

The training data used in the Hallo3 (Cui *et al.*, 2024) model setup consists of a large-scale collection of talking-head videos with speech-driven dynamics, harvested from publicly available videos and curated to provide high-quality facial motion aligned with audio. The original publication reports that the corpus contains 134 hours of raw portrait-video clips used to train a video generation model for dynamic, realistic portrait animation. We use around 87.5 hours after pre-processing.

Internal Talking Face Dataset

In addition to the public datasets above, we use an internal privately-collected audio-visual corpus in our lip-sync project. The dataset consists of around 500 hours of high-resolution recordings of consenting participants speaking in English, in a controlled studio environment, with synchronized close-up facial video from various angles and clean speech audio. Due to privacy and licensing constraints, this corpus is not publicly released and we omit detailed dataset statistics.

2.1.3 Summary of Datasets

We summarise the datasets we used in Table 2.1, including the dataset name, language, number of clips and hours, and whether labels were used.

Dataset	Language	Clips	Hours	Labels
<i>Speech Recognition</i>				
AVSpeech (Ephrat <i>et al.</i> , 2018)	English	721,344	1,333	✗
	Portuguese	180,542	337	✗
	Spanish	110,755	204	✗
	French	64,609	117	✗
	Italian	37,460	68	✗
	Mandarin	36,822	65	✗
VoxCeleb2 (Chung <i>et al.</i> , 2018)	English	617,962	1,323	✗
LRS3 (Afouras <i>et al.</i> , 2018b)	English	166,053	438	✓*
Multilingual TEDx (Salesky <i>et al.</i> , 2021)	Portuguese	46,093	73	✓
	Spanish	38,459	62	✓
	French	49,205	72	✓
	Italian	22,809	40	✓
CMLR (Zhao <i>et al.</i> , 2019)	Mandarin	102,048	86	✓
<i>Lip Synchronization</i>				
HDTF (Zhang <i>et al.</i> , 2021)	English	681	15.8	✗
FEED (Drobyshev <i>et al.</i> , 2024)	N/A	1,411	6.5	✗
NeRsemble (Kirschstein <i>et al.</i> , 2023)	English	6,045	7.5	✗
CelebV-Text (Yu <i>et al.</i> , 2023)	Multiple	18,262	279	✗
CelebV-HQ (Zhu <i>et al.</i> , 2022)	Multiple	5,829	65	✗
RAVDESS (Livingstone and Russo, 2018)	English	2,396	1.4	✗
Hallo3 (Cui <i>et al.</i> , 2024)	English	29,735	87.5	✗
Internal Talking Face Dataset	English	83,743	500	✗

Table 2.1: **Summary of audio-visual datasets used in our projects.** Datasets are grouped by whether they were used for speech recognition or lip synchronization. For multilingual datasets, only the relevant language subsets used are listed. *Used both with and without labels depending on the pipeline stage.

2.2 Data Pre-processing

2.2.1 Audio Pre-Processing

We use raw waveform audio as input, without applying traditional feature extraction methods such as spectrograms, mel-frequency cepstral coefficients (MFCCs), or filterbanks. All audio is resampled to 16 kHz, and for speech recognition projects utterances spanning longer than 24 seconds are split into smaller clips. We normalise to the range $[-1, 1]$ and do not apply any further pre-processing. This minimal pre-processing pipeline allows the model to learn directly from the waveform. An example of a waveform is shown in Figure 2.1.

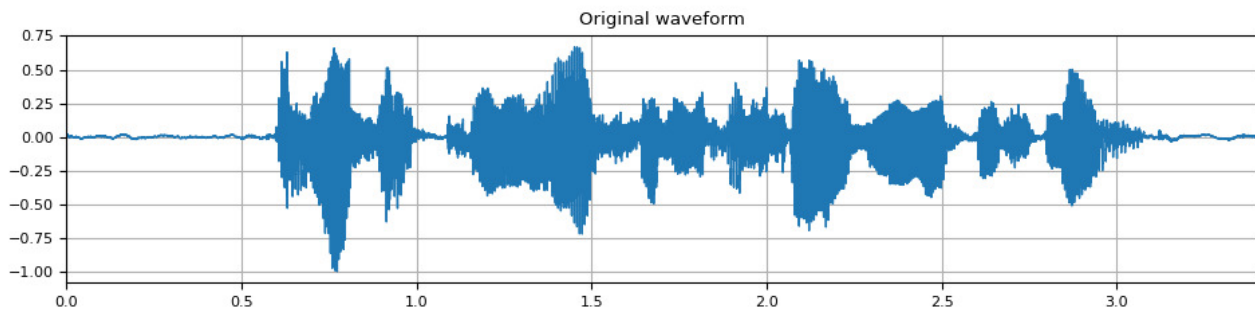


Figure 2.1: **An example of a waveform.** Image adapted from (PyTorch Team, 2024)

2.2.2 Video Pre-Processing

Speech Recognition

For the visual modality, we follow the pre-processing procedure from (Ma *et al.*, 2022). Similarly to the audio pre-processing, utterances spanning longer than 24 seconds are split into smaller clips.

We begin with face detection on each video frame to isolate the speaker’s face region using the RetinaFace method (Deng *et al.*, 2020). RetinaFace predicts the coordinates of the face and facial landmarks. We extract bounding boxes around each subject’s face for each frame, which are used to crop the videos. We then apply the Face Alignment Network (FAN) (Bulat and Tzimiropoulos, 2017) to standardize the orientation of each face across frames. FAN predicts

68 facial landmarks, corresponding to key points such as the eyes, nose, mouth, and jawline. To reduce frame-level noise and motion jitter, we compute the average landmark positions over a 12-frame temporal window. This smoothed set of landmarks is then used to perform an affine transformation to remove translation and scaling variations, aligning each frame to a neutral reference frame template. We then crop a 96×96 region around the speaker's mouth, and convert frames to grayscale. During data loading, we standardise pixel values by subtracting the dataset mean and dividing by the standard deviation. Figure 2.2 illustrates the speech recognition video pre-processing pipeline, while Figure 2.3 shows two example identities before and after processing.

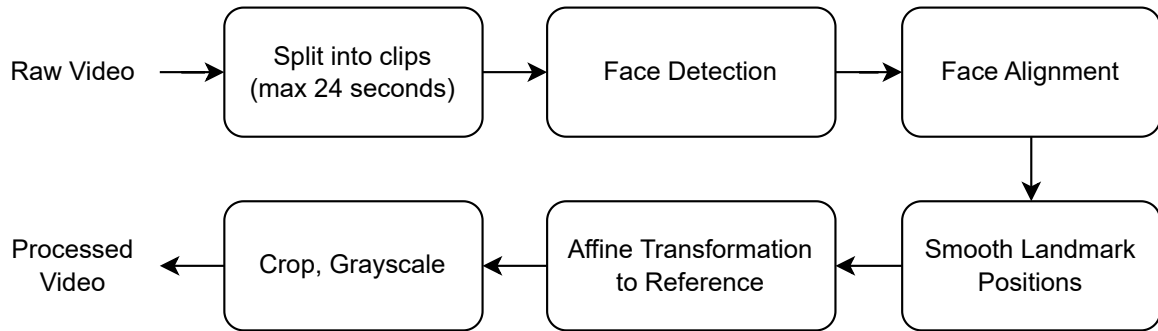


Figure 2.2: **Video pre-processing pipeline for speech recognition.** Overview of the stages applied to each input video.

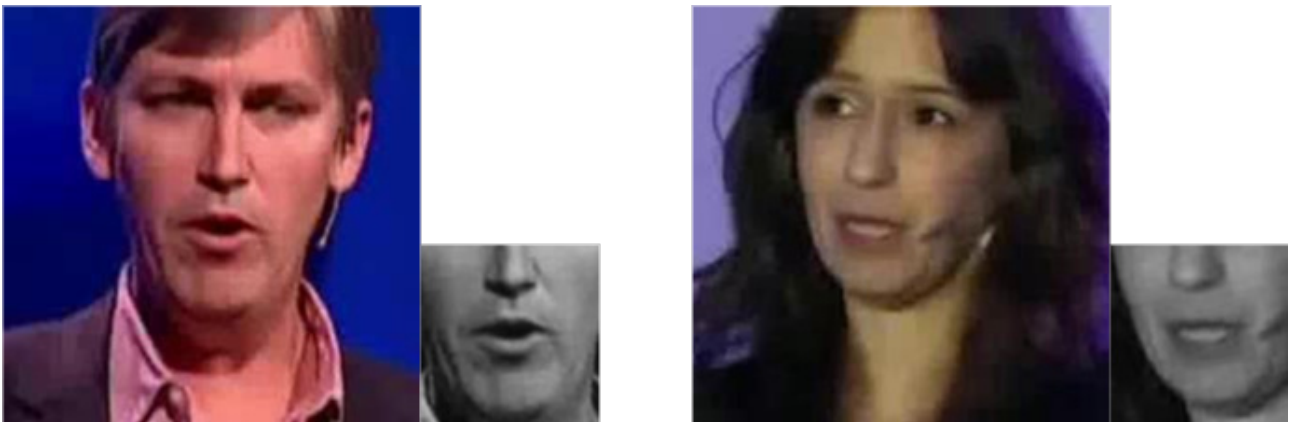


Figure 2.3: **Examples of speech recognition video pre-processing.** Example frames from two different videos before and after processing.

Lip Synchronization

We pre-process all videos used in our lip synchronization experiments using the following pipeline. First, we standardise all clips to 25 frames per second. We discard low-quality videos based on a quality score computed using HyperIQA (Su *et al.*, 2020). We discard segments in which more than one face is detected, or where the speaker’s head moves outside the frame. We detect scene changes and split videos into multiple clips using PySceneDetect (Castellano, 2014). Active-speaker detection is applied using LightASD (Liao *et al.*, 2023) to retain only portions in which the person is speaking, removing silent intervals. Finally, we estimate landmarks and poses using face-alignment (Bulat and Tzimiropoulos, 2017) and crop videos around the facial region to a resolution of 512×512 . Figure 2.4 illustrates the lip synchronization video pre-processing pipeline, with example clips after processing shown in Figure 2.5.

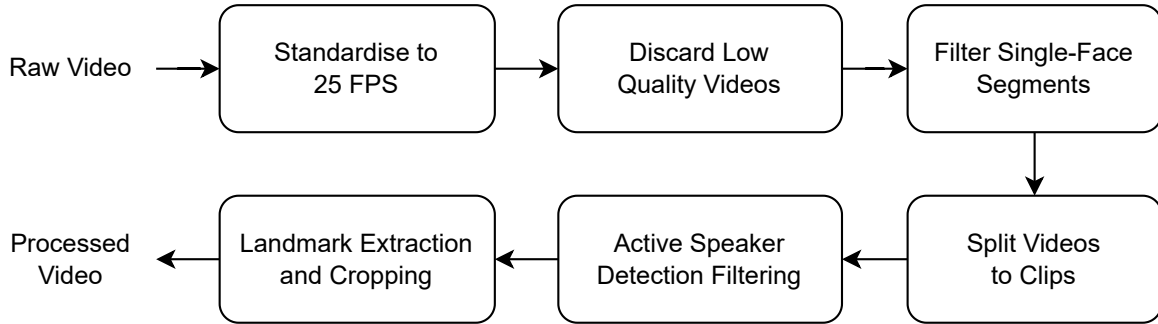


Figure 2.4: **Video pre-processing pipeline for lip synchronization.** Overview of the stages applied to each input video.



Figure 2.5: **Examples of lip synchronization video pre-processing.** Example frames from three different videos after processing.

2.2.3 Text Pre-Processing

We perform text tokenization to pre-process our text labels in our speech recognition works. Text tokenization is the process of converting raw text into smaller units, or tokens, that can be handled by machine learning models. Since most neural networks operate on sequences of discrete symbols, tokenization provides a crucial bridge between natural language and numerical representations.

Two common strategies are character-level and subword-level tokenization. Character-level tokenization treats each character as a token, leading to small vocabularies and natural handling of out-of-vocabulary cases, but longer sequences. This is particularly suitable for languages such as Mandarin, where the combination of characters themselves carry semantic meaning. Subword-level tokenization, on the other hand, balances between characters and words: frequent words remain whole while rare ones are decomposed into smaller units, keeping sequence lengths manageable while preserving generalization.

In our experiments, we used SentencePiece (Kudo and Richardson, 2018) for subword tokenization in all the languages we use, with the exception of Mandarin where we adopted a character-level vocabulary.

2.3 Neural Architectures

This section introduces the core architectures and fundamental building blocks used throughout this thesis: Fully Connected Neural Networks (FCNNs), Convolutional Neural Networks (CNNs), Residual Networks (ResNets), U-Nets and Transformers.

2.3.1 Fully Connected Neural Networks

Fully Connected Neural Networks (FCNNs), recently also known as multilayer perceptrons (MLPs), represent one of the earliest and most fundamental classes of artificial neural networks.

A FCNN is composed of an input layer where the data is received, one or more hidden layers, and an output layer that matches the output dimensionality of the task at hand and outputs the network's prediction. Each layer consists of a set of neurons, where every neuron is connected to every other neuron in the previous and next layer. Nonlinear activation functions are applied to the output of each neuron in the hidden layers, enabling the network to model complex nonlinear relationships. This dense connectivity allows the network to learn arbitrary nonlinear functions, provided it has sufficient capacity and data. A figure of a basic neural network with two hidden layers is shown in Figure 2.6.

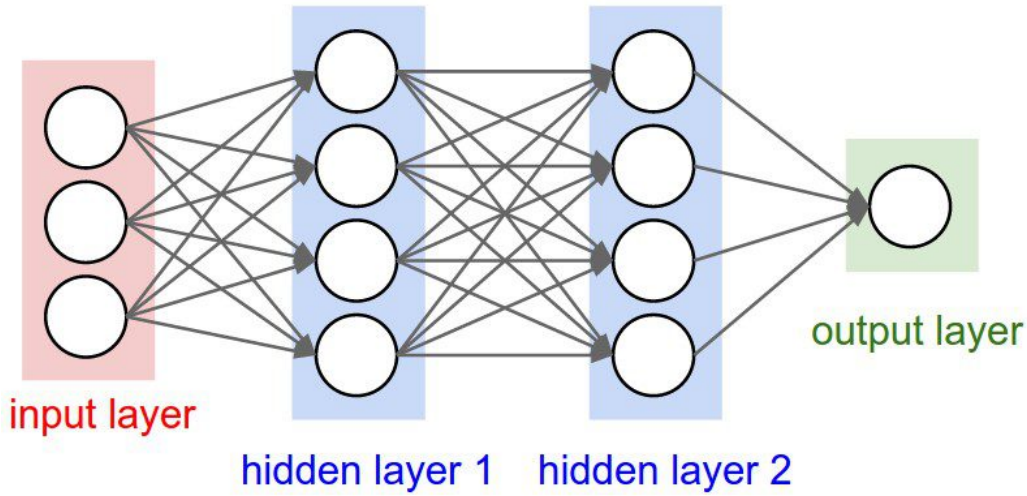


Figure 2.6: **A simple Fully Connected Neural Network (FCNN).** It consists of two hidden layers and an output layer with a single output neuron. Each neuron is connected to all neurons in the previous and next layer. Figure from (Karpathy *et al.*, 2016).

Mathematical Formulation

The mathematical formulation of a single fully connected layer is given by:

$$\mathbf{y} = a(\mathbf{W}^\top \mathbf{x} + \mathbf{b}) \quad (2.1)$$

where $\mathbf{x} \in \mathbb{R}^d$ is the input vector, $\mathbf{W} \in \mathbb{R}^{d \times n}$ is the weight matrix, $\mathbf{b} \in \mathbb{R}^n$ is the bias vector, $\mathbf{y} \in \mathbb{R}^n$ is the output vector, and $a(\cdot)$ is a nonlinear activation function applied element-wise. Without nonlinear activations, the stacked layers would collapse into one big linear function, limiting what the network can learn. Rectified Linear Unit (ReLU) (Agarap, 2018) is a very

popular activation function used by many state-of-the-art architectures which we also employ in our works, as it often works better in deeper networks compared to activations such as sigmoid (Rumelhart *et al.*, 1986) or tanh (LeCun *et al.*, 1998). It is defined as:

$$a(z) = \max(0, z) \tag{2.2}$$

The weights $\mathbf{W}$ and biases $\mathbf{b}$ of all layers are the parameters of the network that must be learned, and are updated during training via the backpropagation algorithm (Rumelhart *et al.*, 1986). The network is trained to minimise its prediction error against a specified ground truth, using pre-defined loss metrics.

While fully connected neural networks (FCNNs) are rarely sufficient on their own for complex predictive tasks, their simplicity and ability to approximate arbitrary nonlinear functions make them a foundational building block in deep learning. They are frequently integrated into more sophisticated architectures, as we will explore in the following sections.

2.3.2 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a fundamental deep learning architecture widely used in computer vision (LeCun, Bengio, *et al.*, 1995). Unlike FCNNs, CNNs are specifically designed to exploit the spatial and local structure of data such as images, video or audio. This is accomplished using weight sharing across spatial regions, localized filter operations, and dimensionality reduction techniques, allowing the model to efficiently capture local patterns. These design choices make CNNs particularly effective for high-dimensional inputs where local structures are important.

A typical CNN is composed of multiple convolutional layers, often followed by nonlinear activation functions, pooling layers for downsampling, and fully connected layers at the output. Each convolutional layer applies a set of learnable filters (or kernels) to its input, capturing patterns such as edges, textures, or more abstract visual features depending on the depth of the network.

Convolutional Layers

Convolutional layers operate by applying the convolution operation to their input. This involves sliding learnable filters, known as kernels, across the spatial dimensions of the input using shared weights. At each spatial location, the kernel computes a weighted sum of the input values, producing a response that forms part of an output feature map. The resulting feature maps, one per kernel, collectively form the new channel dimension of the output tensor. Figure 2.7 shows the convolution operation. While convolution operations can be defined in 1D, 2D, or 3D depending on the nature of the data, we focus on the 2D case in our explanations to maintain clarity and avoid redundancy.

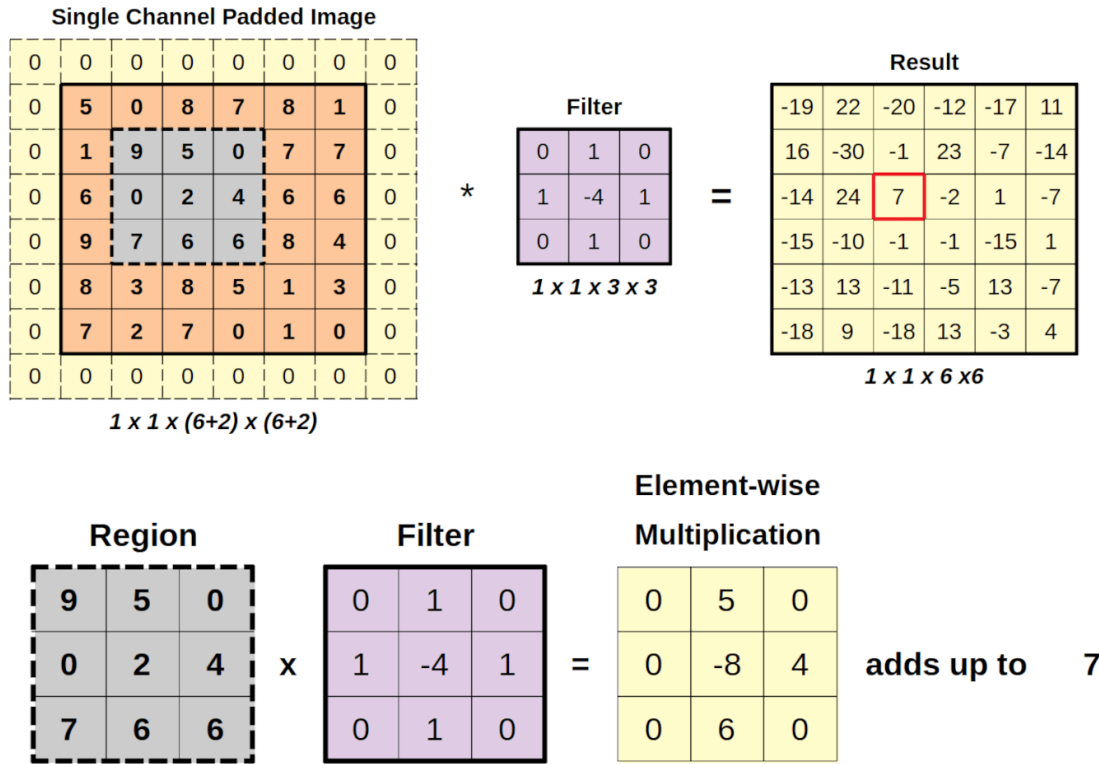


Figure 2.7: **Example of a 2D convolution.** A padding of 1 is applied to the input, shown as yellow cells with value 0. Each 3×3 region of the padded input is multiplied element-wise with the 3×3 kernel, and the results are summed to produce a single output value. Figure adapted from (Godoy, 2021).

The core operation of a 2D convolution can be described as:

$$(I * K)(i, j) = \sum_m \sum_n \sum_c K(m, n, c) \cdot I(i + m, j + n, c) + b \quad (2.3)$$

where I is the input feature map or image at position (i, j) , K is the kernel of size (m, n) , c is an index to the input's channel dimension, and b is the kernel's bias. This operation slides each filter across the spatial dimensions of the input and over its entire channel depth, generating a new feature map that highlights certain structures. Padding and stride parameters also play a role in the convolution, effectively changing the way the convolutions are applied as well as the resulting output size. A general formula for computing the output dimension D_{out} , given an input dimension D_{in} , kernel size K , padding P and stride S is:

$$D_{\text{out}} = \left\lfloor \frac{D_{\text{in}} + 2P - K}{S} + 1 \right\rfloor \quad (2.4)$$

Parameter sharing makes CNNs a very computationally efficient architecture, since the same filter is applied across all spatial locations, significantly reducing the number of parameters compared to FCNNs and improving generalization. Additionally, by stacking multiple convolutional layers, each layer of the network has an increasingly wider receptive field. This allows the network to build hierarchical representations — from low-level edges and textures in early layers to high level object parts or phonetic structures in deeper layers.

Pooling Layers

Pooling layers perform non-learnable operations to reduce the spatial dimensions of feature maps while retaining the most important information. This downsampling operation helps reduce computational complexity, control overfitting, and increase the receptive field. The most common pooling operation is max pooling, which outputs the maximum value within a local window. Another variant is average pooling, which computes the mean of the values in the window. Although modern architectures sometimes replace pooling with strided convolutions, it remains a widely used and conceptually important component in CNNs. Figure 2.8 shows an example of the max pooling operation.

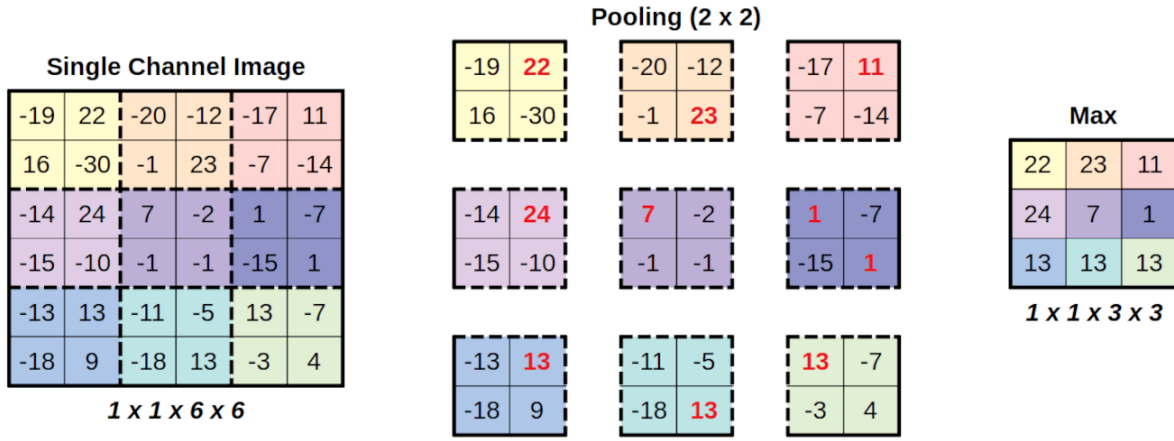


Figure 2.8: **Example of max pooling.** The figure shows a 2×2 max pooling with stride 2, where only the maximum value in each window is kept. Figure from (Godoy, 2021).

Transposed Convolutions

Transposed convolutions, also known as deconvolutions, are used to increase the spatial resolution of the input feature maps. They are commonly used in the decoder paths of encoder-decoder architectures such as U-Net. Unlike simple upsampling methods like interpolation, transposed convolutions are learnable operations that allow the network to learn how to up-sample in a task-specific manner. They work by inserting zeros between input elements and then applying a standard convolution, effectively increasing the output size. Figure 2.9 shows an example of a transposed convolution for a 2×2 image.

Batch Norm

Batch normalization is a method used to stabilize and accelerate the training of deep neural networks by normalizing the activations within a mini-batch (Ioffe and Szegedy, 2015). For each batch, it normalizes the activations of the layer by subtracting the batch mean and dividing by the batch standard deviation, then scales and shifts the result using learnable parameters γ

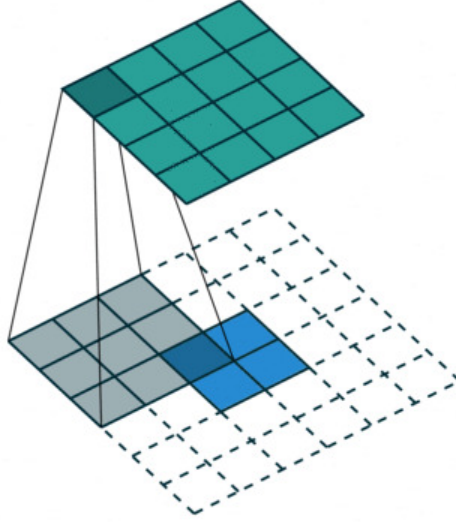


Figure 2.9: **Illustration of a transposed convolution.** The figure shows up-sampling of a 2×2 input to a 4×4 output using a 3×3 kernel with stride 1. Zeros are inserted to expand the input, and then convolved. Figure from (Dumoulin and Visin, 2018).

and β :

$$\hat{\mathbf{x}} = \frac{\mathbf{x} - \boldsymbol{\mu}}{\sqrt{\boldsymbol{\sigma}^2 + \epsilon}} \quad (2.5)$$

$$\mathbf{y} = \gamma \hat{\mathbf{x}} + \beta \quad (2.6)$$

Here, $\mathbf{x}$ is the input activation, $\hat{\mathbf{x}}$ is the normalized activation, $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are the mean and standard deviation computed over the mini-batch, $\mathbf{y}$ is a scaled version of the normalized activation and ϵ is an arbitrarily small constant for numerical stability. At inference time, batch normalization does not use batch statistics. Instead, to ensure consistent behavior across inputs, it uses the running average estimates of the mean $\bar{\boldsymbol{\mu}}$ and variance $\bar{\boldsymbol{\sigma}}^2$ that were accumulated during training over all the batches, with an update rate α :

$$\bar{\boldsymbol{\mu}} \leftarrow (1 - \alpha) \cdot \bar{\boldsymbol{\mu}} + \alpha \cdot \boldsymbol{\mu} \quad (2.7)$$

$$\bar{\boldsymbol{\sigma}}^2 \leftarrow (1 - \alpha) \cdot \bar{\boldsymbol{\sigma}}^2 + \alpha \cdot \boldsymbol{\sigma}^2 \quad (2.8)$$

Batch normalization helps reduce internal covariate shift, improves gradient flow, and also has a mild regularization effect, often reducing the need for dropout in convolutional layers. In

CNNs, batch normalization is typically applied after convolution and before the nonlinearity.

2.3.3 ResNets

Residual Networks (ResNets) (He *et al.*, 2016) introduced the concept of residual (or skip) connections, which allow the input to bypass one or more layers and be added back to the transformed features. This design addresses the problem of vanishing gradients in deep networks by enabling direct gradient flow and stabilising training. Instead of learning a full mapping $\mathcal{H}(\mathbf{x})$, ResNets learn the residual $\mathcal{F}(\mathbf{x})$, and add it back to the input:

$$\mathcal{H}(\mathbf{x}) = \mathcal{F}(\mathbf{x}) + \mathbf{x} \quad (2.9)$$

A typical residual block consists of two or three convolutional layers with batch normalization and ReLU activations, followed by an element-wise addition with the block's input. Figure 2.10 shows a basic ResNet block with two convolutional layers.

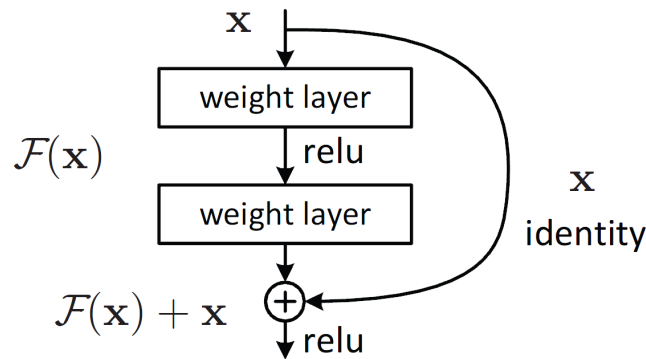


Figure 2.10: **Example of a basic residual block.** The block consists of two layers, ReLU activations and a skip connection. Figure from (He *et al.*, 2016).

ResNet architectures come in various depths, such as ResNet-18 and ResNet-34 as seen in Figure 2.11, where the number indicates the total number of layers. Their stability and hierarchical structure make them well-suited for processing spatial and temporal data, including both video and audio. In our speech recognition works, we use ResNet-18 backbones with 1D and 3D convolutional input stems for audio and video inputs respectively. These modality-specific stems are used to project the raw input into a feature space compatible with the residual network.

34-layer residual

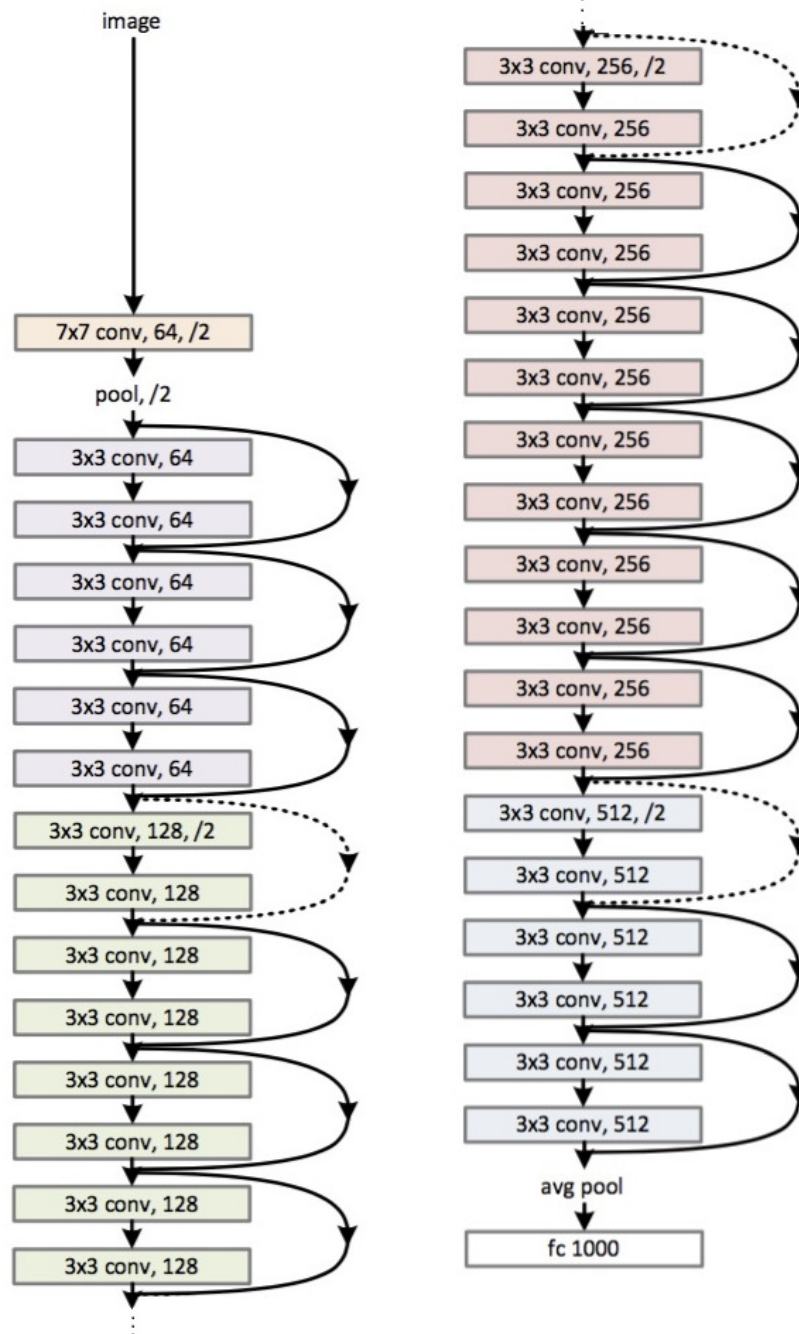


Figure 2.11: **Architecture of the ResNet-34.** The figure shows all of its 34 layers and skip connections. Figure adapted from (He *et al.*, 2016).

2.3.4 U-Nets

The U-Net is a convolutional neural network architecture originally introduced for biomedical image segmentation (Ronneberger *et al.*, 2015), but it has since become widely adopted across computer vision and speech domains for tasks requiring dense, structured output. The name comes from the network’s U-shape formed by an encoder-decoder structure with skip connections that allow fine-grained details to be preserved during upsampling. We use a U-Net architecture for one of our model variants in our lip synchronization work.

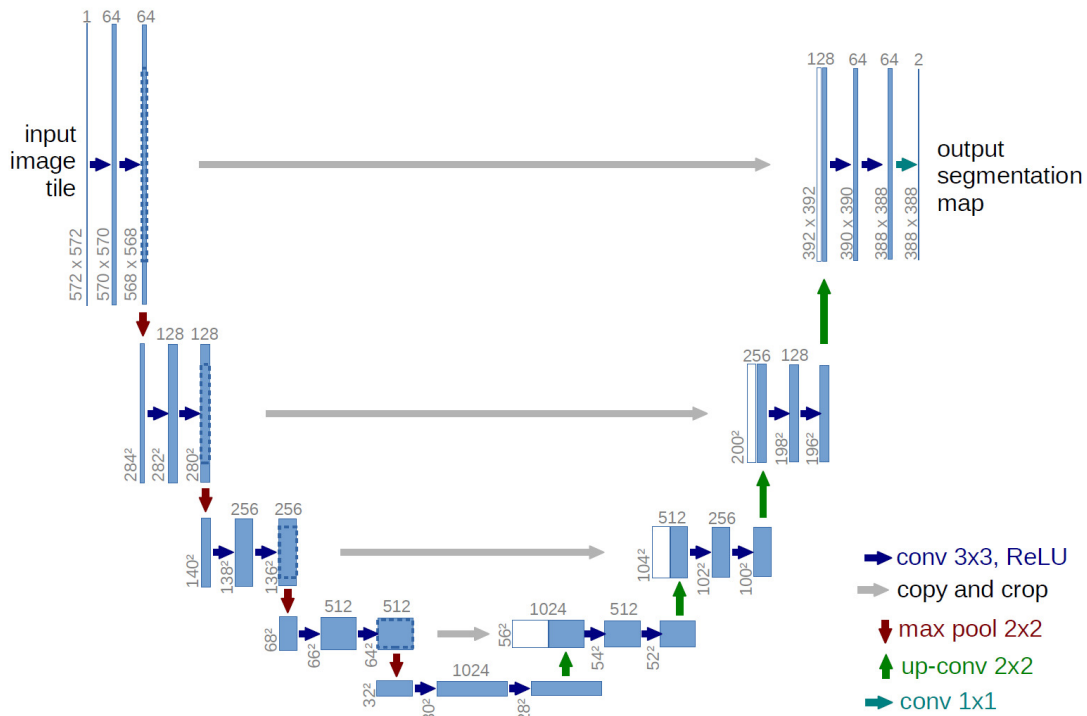


Figure 2.12: **Illustration of the U-Net architecture.** It consists of an encoder-decoder structure with skip connections. Downsampling is performed via strided convolution or pooling; upsampling uses transposed convolutions or interpolation. Figure from (Ronneberger *et al.*, 2015).

The encoder compresses the input through consecutive convolution and downsampling layers, capturing increasingly abstract, high-level features. The decoder then progressively upsamples the feature maps to recover spatial resolution and generate a dense prediction, symmetrically reversing the encoder operations. At each stage in the decoder, feature maps from the corresponding encoder layer are concatenated via skip connections. These skip connections restore spatial details that may be lost during the downsampling operations, enabling precise localisa-

tion in the output. A figure of the architecture is shown in Figure 2.12.

Unlike classification-focused CNNs such as ResNet, U-Nets produce output maps that match the input dimensions, making them ideal for pixel-wise prediction tasks. Variants of the U-Net often include residual blocks in the encoder or decoder, or use attention-based mechanisms in the skip connections to enhance feature fusion.

2.3.5 Transformers

Transformers are a class of neural network architectures originally proposed for natural language processing (Vaswani *et al.*, 2017), which have since become foundational in a wide range of domains, including computer vision, audio modeling, and multimodal learning. Unlike recurrent architectures such as RNNs (Elman, 1990), LSTMs (Hochreiter and Schmidhuber, 1997) or GRUs (Cho *et al.*, 2014) which process inputs sequentially, transformers remove the need for recurrence and instead rely on self-attention mechanisms to capture dependencies across input elements. This enables efficient parallel modeling of both local and global context. We use the transformer architecture in all our works in this thesis. Figure 2.13 shows the transformer architecture and its components.

Architecture Overview

The transformer architecture is composed of an encoder and a decoder, each consisting of a stack of identical layers, and capable of handling input and output sequences that may differ in length. The inputs are first projected into embeddings, and positional embeddings are added to preserve the order of the input elements. Each encoder layer includes a multi-head self-attention mechanism and a feedforward network, both followed by residual connections and layer normalization. Decoder layers add a second attention block that performs cross-attention with the encoder’s output, enabling the decoder to create more informed predictions. During training, masking is applied to the decoder only to prevent attention to future tokens.

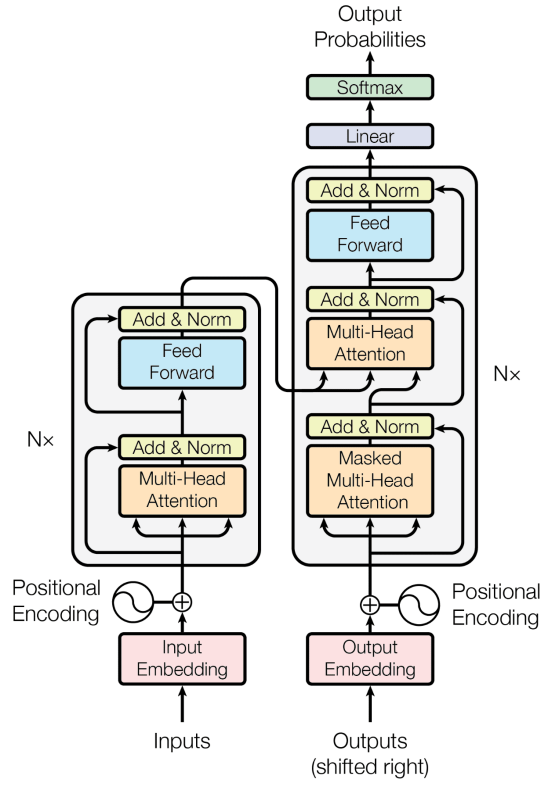


Figure 2.13: **Illustration of the transformer architecture.** The figure shows the encoder-decoder structure, with their various components. Figure from (Vaswani *et al.*, 2017).

Attention & Multi-Head Attention

The core building block of the transformer is the attention mechanism, and more specifically multi-head attention, which allows the model to dynamically weigh different parts of the input sequence when computing representations. Given an input sequence of length T of embeddings $\mathbf{X} \in \mathbb{R}^{T \times d_{model}}$, the attention mechanism computes a weighted sum of values $\mathbf{V}$ using learned queries $\mathbf{Q}$ and keys $\mathbf{K}$:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right) \mathbf{V} \quad (2.10)$$

where $\mathbf{Q}, \mathbf{K} \in \mathbb{R}^{T \times d_k}$, $\mathbf{V} \in \mathbb{R}^{T \times d_v}$ are linear projections of the input.

To improve representational power the authors introduced multi-head attention, which splits the attention mechanism into multiple heads with separate projections. The outputs of these heads are then concatenated and projected again. This allows the model to capture unique

information from different representation subspaces for each head, which is not possible with a single head. Figure 2.14 shows both single-head and multi-head attention. Multi-head attention can be formulated as:

$$\text{MultiHeadAttention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Concat}(\text{head}_1, \dots, \text{head}_h) \mathbf{W}^O \quad (2.11)$$

$$\text{where head}_i = \text{Attention}(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V) \quad (2.12)$$

where $\mathbf{W}_i^Q$, $\mathbf{W}_i^K$, $\mathbf{W}_i^V$ and $\mathbf{W}^O$ are learnable projection weights.

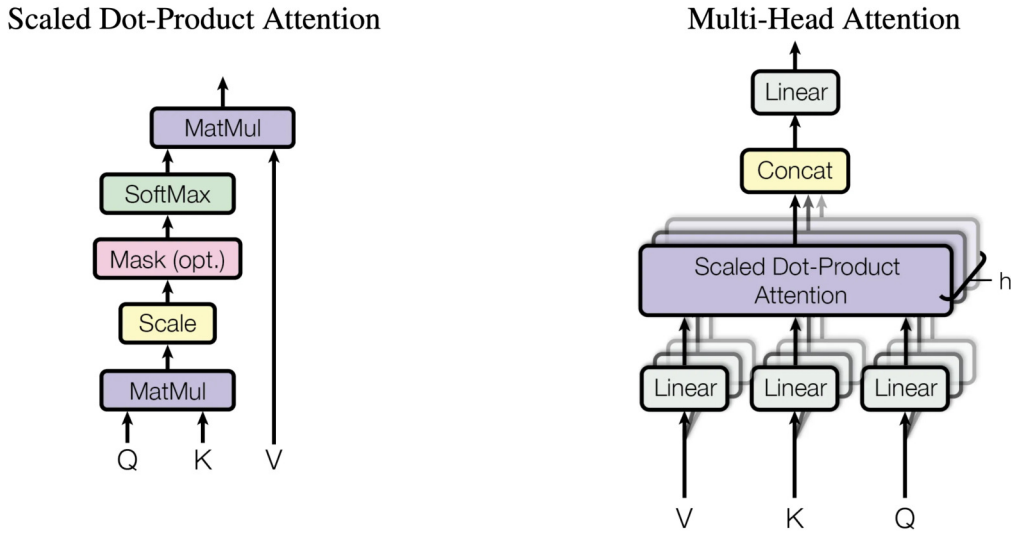


Figure 2.14: **Single and multi-head attention mechanisms.** Multi-head attention allows the model to capture diverse contextual relationships by projecting the input into different representation subspaces. Figure from (Vaswani *et al.*, 2017).

Positional Embeddings

Transformers are permutation-invariant and therefore require positional information to model sequence order. This is achieved by adding positional embeddings to the input token representations at the base of the model. These embeddings encode either absolute or relative position within the sequence. Common approaches include fixed sinusoidal encodings and learned, trainable positional embeddings.

Layer Norm

Layer Normalization (LayerNorm) (Ba *et al.*, 2016) is a normalization technique commonly used in transformer architectures that improves gradient flow and stabilizes and accelerates training. Unlike batch normalization, which normalizes features across a batch of samples, LayerNorm normalizes across the feature dimension of each individual input independently, and is invariant to batch size. LayerNorm is typically applied before or after each sub-layer in a transformer block in combination with residual connections. Given an input vector $\mathbf{x} \in \mathbb{R}^d$, LayerNorm computes the normalized output as:

$$\hat{\mathbf{x}} = \frac{\mathbf{x} - \boldsymbol{\mu}}{\sqrt{\boldsymbol{\sigma}^2 + \epsilon}} \cdot \boldsymbol{\gamma} + \boldsymbol{\beta} \quad (2.13)$$

where $\boldsymbol{\mu}$ is the mean of the feature dimension, $\boldsymbol{\sigma}$ the standard deviation, $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ are learned parameters and ϵ is a small constant for numerical stability.

Feed Forward Layers

In transformer architectures, each layer includes a position-wise feedforward network (FFN) that operates independently on each token of the sequence. This block consists of two fully connected layers with a ReLU activation in between. These layers increase the model's capacity by enabling complex, nonlinear transformations at each token location, independent of sequence order.

2.4 Representation Learning

This section reviews the main families of representation learning that provide background for our work: supervised, unsupervised, and self-supervised learning.

2.4.1 Supervised Learning

Supervised learning is one of the most fundamental and widely used methodologies in machine learning. A model f is trained on inputs $\mathbf{x}$ accompanied by a ground-truth label $\mathbf{y}$, by minimising a pre-defined loss function $L(\mathbf{y}, \hat{\mathbf{y}})$ that measures how close the predictions $\hat{\mathbf{y}} = f(\mathbf{x})$ are to the labels. It is usually used for two broad categories of tasks: regression (predicting continuous values) and classification (predicting distinct classes).

In audio-visual human behaviour modelling, supervised learning has been used for tasks such as speech recognition, speaker verification, and emotion classification (Afouras *et al.*, 2018a; Graves *et al.*, 2013; Petridis *et al.*, 2018; Snyder *et al.*, 2018; Afouras *et al.*, 2022; Goncalves *et al.*, 2024). When given plenty of annotated data, supervised models can extract task-specific representations and achieve state-of-the-art results. However, the success of supervised methods is limited by the availability of high-quality labelled data, which in audio-visual domains – among others – is often scarce and expensive to collect. Furthermore, training in a fully supervised fashion can lead to overfitting and poor generalisation when there is a domain shift – in cases such as language, accents, demographics, or recording conditions.

These challenges motivated researchers to find ways to leverage the vast amounts of unlabelled data that exists in the wild, in order to train robust, accurate models. Approaches such as unsupervised and self-supervised learning have been a result of this motivation, which we will discuss in the following sections. Even when not used for end-to-end training, supervised learning remains crucial for downstream evaluation and for fine-tuning models that were pre-trained using non-supervised methodologies.

2.4.2 Unsupervised Learning

Unsupervised learning aims to extract meaningful patterns and structure from unlabeled data (Bengio *et al.*, 2013). Clustering techniques such as k-means and Gaussian mixture models group similar data points, while dimensionality reduction methods like PCA and t-SNE project high-dimensional data into a lower-dimensional space. In audio-visual domains, these methods

have been used to uncover statistical dependencies between modalities (Ngiam *et al.*, 2011). Figure 2.15 shows a high-level comparison between unsupervised and supervised learning.

With the advent of deep learning, powerful models such as autoencoders have become a central unsupervised technique. These models learn to encode inputs into compressed latent spaces from which the original data can be approximately reconstructed, encouraging the discovery of compact representations (Goodfellow *et al.*, 2016). Autoencoders form the basis for many modern latent generative models, by enabling more efficient and robust training and inference of diffusion and flow-matching models in latent space, rather than directly in pixel or waveform space, where direct modeling is often infeasible (Rombach *et al.*, 2022). This latent generative modeling framework also underpins our lip synchronization work.

Classical unsupervised learning methods typically fall short of supervised approaches in terms of performance, in tasks where labels are available. Consequently, recent advances often integrate unsupervised objectives within self-supervised or hybrid frameworks (Shi *et al.*, 2022), or use unsupervised models as components in more complex systems.

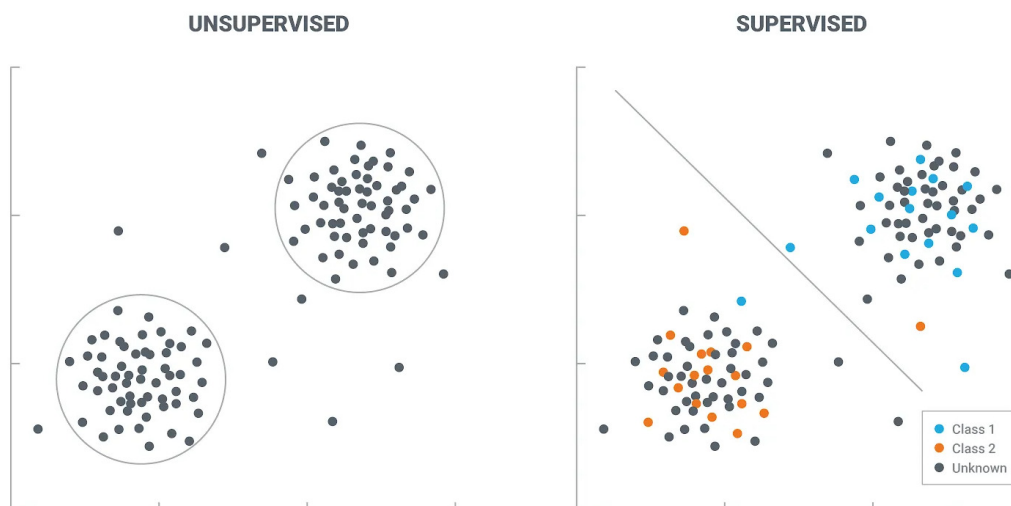


Figure 2.15: **A high-level comparison of unsupervised and supervised learning.** Unsupervised learning groups similar data points into clusters without using labels, while supervised learning uses labels to find boundaries for classification. Figure from (Lawtomated, 2021).

2.4.3 Self-supervised Learning

Self-supervised learning (SSL) aims to combine the strengths of supervised and unsupervised learning. It has emerged as a foundational methodology for representation learning, especially in domains where labelled data is scarce or expensive to obtain. Instead of relying on manually annotated labels, SSL leverages the natural structure and information in the data itself to define pretext tasks with corresponding pseudo-labels. Solving these pretext tasks enables models to learn representations that generalise well to downstream supervised tasks (Jing and Tian, 2020), such as segmentation (Long *et al.*, 2015), object detection (Ren *et al.*, 2016), or image classification (Krizhevsky *et al.*, 2012). Figure 2.16 shows a high level overview of a generic self-supervised learning pipeline.

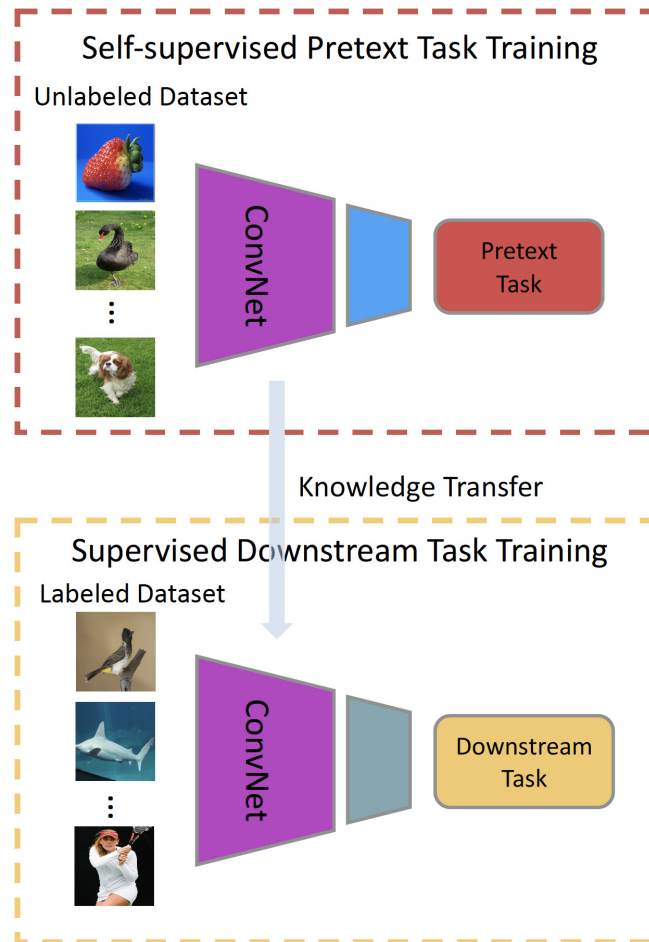


Figure 2.16: **General pipeline of self-supervised learning.** Embeddings are learned by training deep neural networks on a pretext task using pseudo-labels. Once training is complete, the resulting pre-trained model is used to extract embeddings, which are then transferred to downstream tasks that rely on real labels. Figure from (Jing and Tian, 2020).

SSL methods fall into four high-level families: **contrastive**, **generative**, **clustering-based**, and **distillation-based** methods (Balestriero *et al.*, 2023b):

- **Contrastive methods** learn by generating different views of a sample, then distinguishing positive pairs (views from the same sample) from negative pairs (views from different samples) (Oord *et al.*, 2018; He *et al.*, 2020; Chen *et al.*, 2020). These approaches rely heavily on carefully designed data augmentations and large batch sizes or memory banks to construct sufficient negatives.
- **Generative methods** attempt to reconstruct a given input after it undergoes corruption or masking. It is a fundamental training method in many state-of-the-art architectures for natural language and audio-visual modalities (He *et al.*, 2022; Hsu *et al.*, 2021; Devlin *et al.*, 2019).
- **Clustering-based methods** encourage samples to be grouped into discrete clusters in representation space (Caron *et al.*, 2018, 2020). These methods avoid negative sampling by aligning representations with corresponding learned cluster centroids they are close to.
- **Distillation-based methods** bypass the need for negative samples or explicit clustering by leveraging a teacher-student architecture, where the student learns to predict the teacher’s output (Grill *et al.*, 2020; Caron *et al.*, 2021; Baevski *et al.*, 2022).

In the audio-visual domain, SSL exploits the synchrony between modalities. Pretext tasks such as audio-visual alignment, masked prediction, and cross-modal generation are used to leverage shared structure (Shi *et al.*, 2022; Alwassel and al., 2020; Ma *et al.*, 2021a). Our speech recognition work builds directly on these advances by utilizing and enhancing novel architectures like RAVEn (Haliassos *et al.*, 2023), which extends self-supervised learning methods to cross-modal settings, and integrates ideas from masked learning and distillation. These self-supervised approaches are very effective in speech recognition from talking face videos, where the correlation between audio and visual signals provides a strong supervisory signal without requiring manual annotations or laborious augmentations.

2.5 Generative Modeling

This section introduces the main paradigms of generative modeling that provide background for this thesis: autoencoders, generative adversarial networks, diffusion models, and flow matching models.

2.5.1 Autoencoders

Autoencoders are a class of neural networks designed to learn compact representations of data through unsupervised reconstruction (Hinton and Salakhutdinov, 2006; Bank *et al.*, 2020). Their architecture, shown in Figure 2.17, consists of two main components: an encoder network g_ϕ that maps the input $\mathbf{x} \in \mathbb{R}^d$ to a latent representation $\mathbf{z} \in \mathbb{R}^e$, where usually $e \ll d$, and a decoder network f_θ that attempts to reconstruct the original input, producing $\mathbf{x}' = f_\theta(g_\phi(\mathbf{x}))$. The model is trained to minimise a reconstruction loss between $\mathbf{x}$ and $\mathbf{x}'$, typically using an ℓ_1 or ℓ_2 loss:

$$\mathcal{L}_{\text{recon}} = \|\mathbf{x} - \mathbf{x}'\|^2 \quad (2.14)$$

By compressing the input into a lower-dimensional latent space, autoencoders are encouraged to extract the most important features of the data, essentially performing dimensionality reduction. In the context of audio-visual learning, autoencoders can be applied to each modality. For instance, a visual encoder might map a cropped face sequence to a compact latent embedding, which a decoder can use to reconstruct. As mentioned before, latent representations of data are also frequently used to train large models in latent space, before decoding back to the original space.

Vanilla autoencoders are deterministic and do not offer a probabilistic output that is crucial for many applications like generation. Since they do not impose any structure in the latent space, they also often fail to produce representations that are robust, smooth or easily controllable. This motivates the development of more structured variants such as Variational Autoencoders.

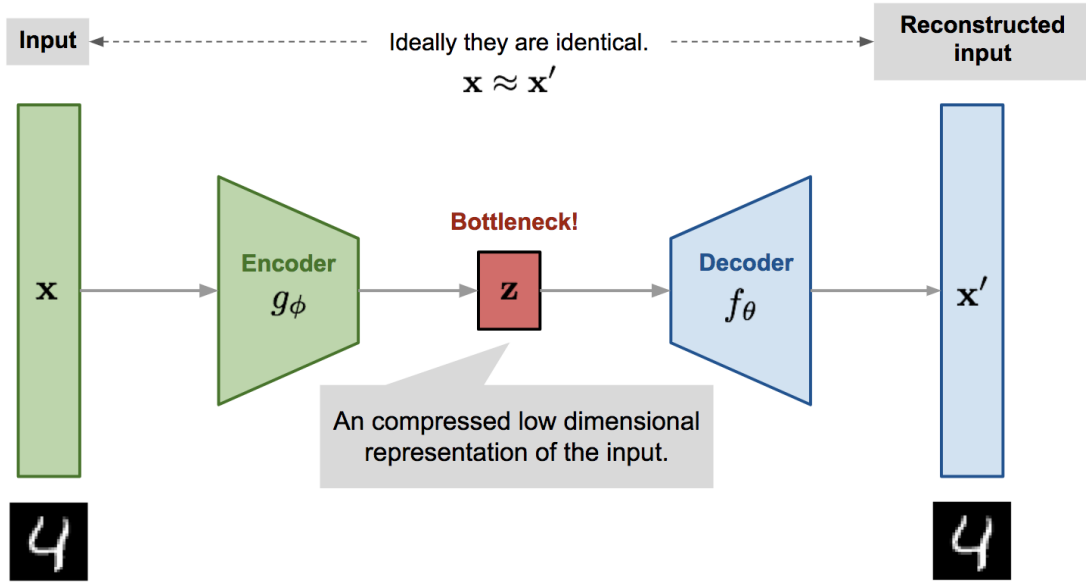


Figure 2.17: **Diagram of a vanilla autoencoder.** The autoencoder takes an input $\mathbf{x}$ into the encoder g_ϕ to produce latent $\mathbf{z}$, and then gives it to decoder f_θ to get the reconstruction $\mathbf{x}'$. Figure from (Weng, 2018).

Variational Autoencoders

Variational Autoencoders (VAEs) extend standard autoencoders by introducing a probabilistic framework for representation learning (Kingma and Welling, 2014). Instead of encoding each input $\mathbf{x}$ to a single point in latent space, VAEs model the latent representation $\mathbf{z}$ as a random variable drawn from an approximate posterior $q_\phi(\mathbf{z}|\mathbf{x})$, typically parameterised as a Gaussian with mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\sigma}^2$. The decoder $p_\theta(\mathbf{x}|\mathbf{z})$ then reconstructs the input from samples of $\mathbf{z}$, as seen in Figure 2.18. One key advantage of VAEs is their ability to generate meaningful interpolations between points in latent space, due to the smoothness and continuity induced by the KL divergence term. This smooth latent structure enables meaningful interpolation and controlled sampling.

Training a VAE involves maximising the Evidence Lower Bound (ELBO) on the marginal log-likelihood $\log p_\theta(\mathbf{x})$:

$$\mathcal{L}_{\text{VAE}} = \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} [\log p_\theta(\mathbf{x}|\mathbf{z})] - D_{\text{KL}}[q_\phi(\mathbf{z}|\mathbf{x}) \parallel p_\theta(\mathbf{z})] \quad (2.15)$$

Here, $p_\theta(\mathbf{z})$ is a prior distribution over the latent space, typically chosen as a standard normal

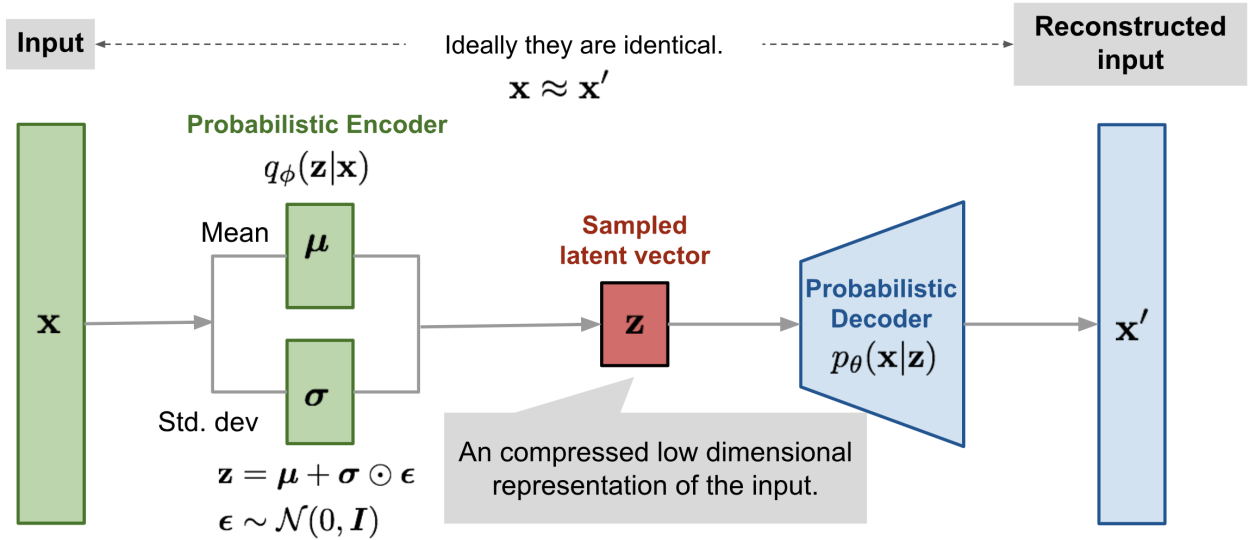


Figure 2.18: **Diagram of a variational autoencoder.** The diagram shows the VAE architecture with the encoder outputting μ and σ , sampling $\mathbf{z}$ via reparameterisation trick, and reconstruction $\mathbf{x}'$ via decoder. Figure from Weng, 2018.

$\mathcal{N}(\mathbf{0}, \mathbf{I})$, and D_{KL} denotes the Kullback-Leibler divergence. The first term encourages accurate reconstruction, while the second regularises the latent space by aligning it with the prior. The reparameterisation trick is crucial for backpropagation through the stochastic sampling step. Instead of sampling $\mathbf{z} \sim \mathcal{N}(\mu, \sigma^2)$ directly, one samples $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and computes $\mathbf{z} = \mu + \sigma \cdot \epsilon$, making the operation differentiable. This formulation effectively separates the stochastic part ϵ from the learnable parameters μ, σ , enabling low-variance gradient estimates through standard backpropagation.

In audio and image synthesis, discrete variants such as VQ-VAE have demonstrated strong performance in modeling raw waveforms and compressed representations (Oord *et al.*, 2017). Latent Diffusion Models (LDMs) (Rombach *et al.*, 2022) utilize VAEs to encode high-dimensional inputs into compact latent spaces, where denoising diffusion is applied. This approach enables faster, memory-efficient, and high-resolution synthesis, and has since been adapted to audio and video domains.

2.5.2 Generative Adversarial Networks

Generative Adversarial Networks (GANs) are a class of generative models that formulate the learning problem as a game between two neural networks: a generator G and a discriminator D (Goodfellow *et al.*, 2014). The generator learns to produce samples $G(\mathbf{z})$ from a random noise vector $\mathbf{z} \sim p_z(\mathbf{z})$, while the discriminator attempts to distinguish real data samples $\mathbf{x} \sim p_{\text{data}}(\mathbf{x})$ from fake ones generated by G .

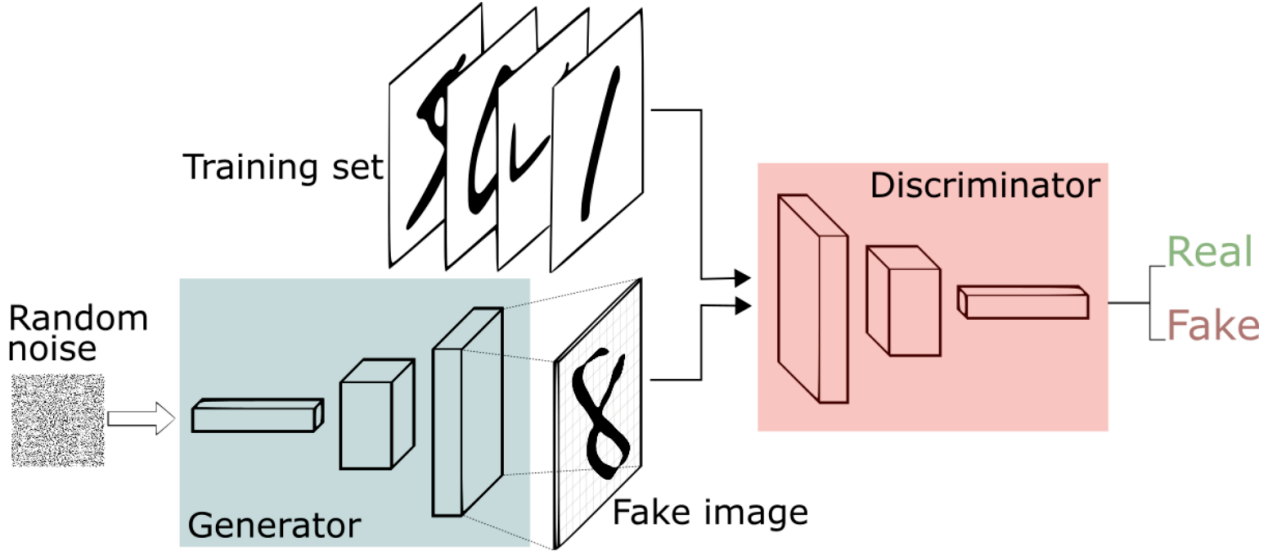


Figure 2.19: **Diagram of a GAN.** The framework shows a Generator producing synthetic images from random noise and a Discriminator trying to distinguish between the real and synthetic samples. Figure from (Buff ML, 2022).

The training objective is formulated as a min-max game between G and D :

$$\mathcal{L}_{\text{GAN}} = \min_G \max_D \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))] \quad (2.16)$$

When trained successfully, the generator learns to produce realistic samples that lie close to the true data distribution. Some generated samples can be seen in Figure 2.20. Unlike likelihood-based models such as VAEs which rely on explicit reconstruction losses, GANs are trained adversarially without reconstructing specific inputs. This allows them to focus on generating realistic samples that resemble the data distribution as a whole, rather than matching individual examples. As a result, GANs often produce sharper and more visually appealing outputs, especially for image and video data.



Conditional GANs

Conditional GANs extend the original GAN framework, in which both G and D receive additional conditioning input $\mathbf{y}$ such as class label, audio, or another image, allowing more controlled generation following the condition. The objective becomes:

$$\mathcal{L}_{\text{cGAN}} = \min_G \max_D \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x} | \mathbf{y})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z}), \mathbf{y} \sim p_{\text{data}}(\mathbf{y})} [\log (1 - D(G(\mathbf{z} | \mathbf{y}) | \mathbf{y}))] \quad (2.17)$$

This approach is particularly relevant in audio-visual modeling, where one modality is used to condition the generation of another, for example generating facial motion conditioned on speech (Vougioukas *et al.*, 2019; Suwajanakorn *et al.*, 2017; Chen *et al.*, 2019). These models aim to generate temporally coherent, identity-preserving face sequences that align with the audio content. Similarly, video-to-video translation methods employ GANs to model motion and content across frames (Wang *et al.*, 2018). An example is shown in Figure 2.21.

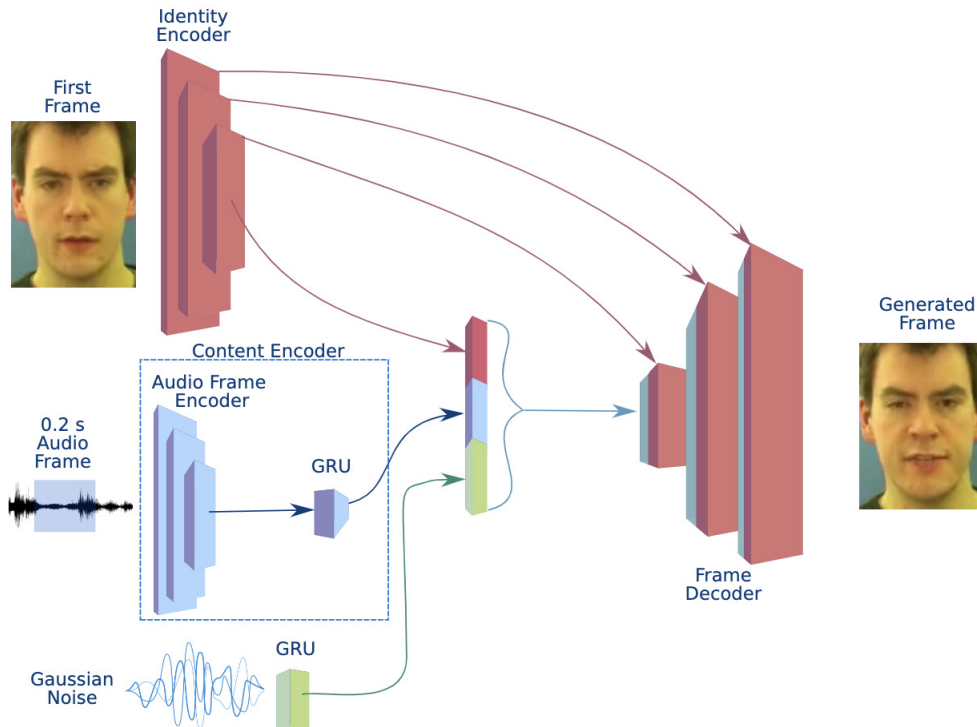


Figure 2.21: **Diagram of a conditional GAN.** The generator is conditioned by extracted audio and identity image features, producing facial motion frames. Figure from (Vougioukas *et al.*, 2019).

Although training stability has improved significantly in recent years, GANs can still present challenges such as mode collapse and sensitivity to hyper-parameters. Stabilisation techniques such as the Wasserstein loss (Arjovsky *et al.*, 2017), spectral normalization (Miyato *et al.*, 2018), and temporal discriminators (Tulyakov *et al.*, 2018) have proven effective, particularly in video and cross-modal settings. Despite these complexities, GANs remain a powerful and widely used tool for generative modelling, especially in high-fidelity synthesis and conditional generation.

2.5.3 Diffusion Models

Diffusion models, in particular denoising diffusion probabilistic models (DDPMs), are a class of generative models that have recently achieved state-of-the-art results across image, audio, and video generation tasks. They define a generative process by reversing a gradual noising process, known as the diffusion process, which transforms data into noise (Ho *et al.*, 2020b). By learning to reverse this process, these models are able to generate realistic samples by denoising from pure noise step by step, as seen in Figure 2.22. Some generated samples from diffusion models are shown in Figure 2.23.

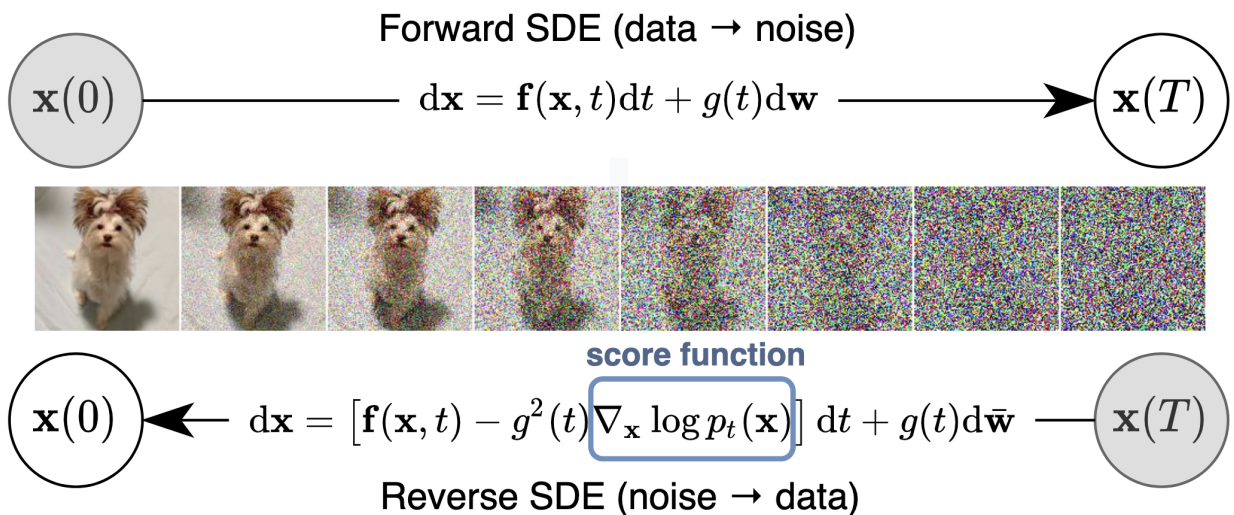


Figure 2.22: A figure illustrating the forward and reverse diffusion process. The input $\mathbf{x}_0$ is progressively corrupted and denoised back to its original form (Song *et al.*, 2021).

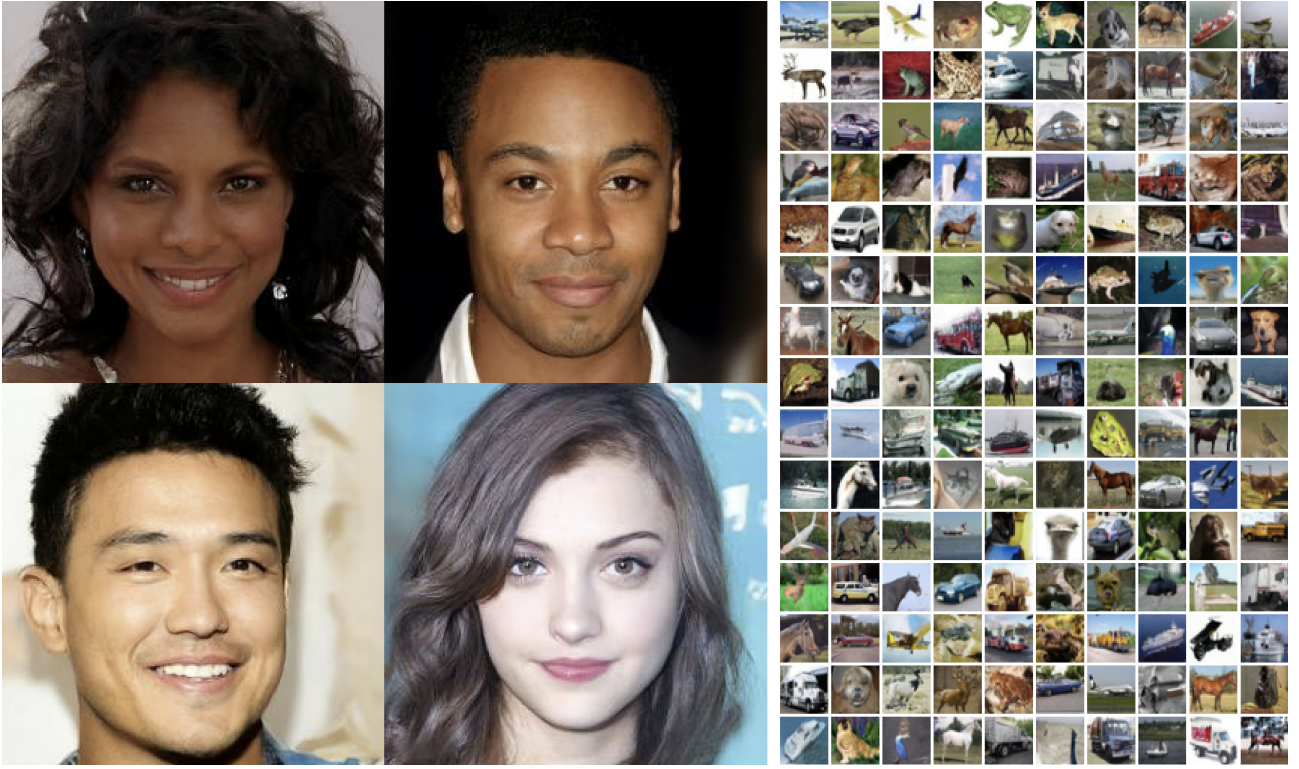


Figure 2.23: **Some generated samples from DDPMs.** Samples are from the CelebA-HQ (Karras *et al.*, 2018) (left) and CIFAR10 (Krizhevsky, 2009) (right) datasets. Figure from (Ho *et al.*, 2020b).

Training Process

A diffusion model consists of two key processes: a forward (noising) process and a reverse (denoising) process. In the forward process, a clean data sample $\mathbf{x}_0$ is progressively corrupted over T timesteps by adding Gaussian noise, forming a sequence of latent variables $\{\mathbf{x}_t\}_{t=1}^T$. This is typically defined as:

$$q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1 - \beta_t} \mathbf{x}_{t-1}, \beta_t \mathbf{I}) \quad (2.18)$$

where β_t is the variance schedule, often linearly or cosine increasing, and $\mathbf{I}$ is the identity matrix. By composing these steps, we get the marginal distribution:

$$q(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t} \mathbf{x}_0, (1 - \bar{\alpha}_t) \mathbf{I}) \quad (2.19)$$

with $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{s=1}^t \alpha_s$. The above closed form allows us to directly calculate $\mathbf{x}_t$ from $\mathbf{x}_0$ without going through the intermediate steps.

The reverse process attempts to learn the inverse of this corruption using a neural network parameterised by θ :

$$p_{\theta}(\mathbf{x}_{t-1} \mid \mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_{\theta}(\mathbf{x}_t, t), \boldsymbol{\Sigma}_{\theta}(\mathbf{x}_t, t)) \quad (2.20)$$

The training objective is derived by minimizing a variational lower bound on the negative log-likelihood. After some mathematics and simplifications (Ho *et al.*, 2020b), the loss becomes:

$$\mathcal{L}_{\text{simple}} = \mathbb{E}_{t, \mathbf{x}_0, \epsilon} [\|\epsilon - \epsilon_{\theta}(\mathbf{x}_t, t)\|^2] \quad (2.21)$$

where ϵ is the noise sampled from $\mathcal{N}(0, \mathbf{I})$, and ϵ_{θ} is the network's prediction. Similarly to GANs (Section 2.5.2), diffusion models can be conditioned on various types of inputs $\mathbf{y}$ such as images or audio. We omit conditioning symbols in most equations for simplicity, conditional variants simply replace $\epsilon_{\theta}(\mathbf{x}_t, t)$ with $\epsilon_{\theta}(\mathbf{x}_t, t \mid \mathbf{y})$.

Sampling Process

Once training is complete, new data samples can be generated by sampling from the standard normal distribution and iteratively denoising using the learned model. Starting from pure Gaussian noise $\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$, the model successively applies the reverse process $p_{\theta}(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$ for $t = T, \dots, 1$ to generate the sample $\mathbf{x}_0$.

A simplified form of the reverse update at each step is given by:

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_{\theta}(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z} \quad (2.22)$$

where $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$ if $t > 1$ else $\mathbf{z} = 0$, and σ_t is a precomputed noise scale that controls the stochasticity of each step. The training and sampling procedures are outlined in Algorithm 2.24.

Algorithm 1 Training

```

1: repeat
2:    $\mathbf{x}_0 \sim q(\mathbf{x}_0)$ 
3:    $t \sim \text{Uniform}(\{1, \dots, T\})$ 
4:    $\epsilon \sim \mathcal{N}(0, \mathbf{I})$ 
5:   Take gradient descent step on
6:    $\nabla_{\theta} \|\epsilon - \epsilon_{\theta}(\sqrt{\alpha_t}\mathbf{x}_0 + \sqrt{1 - \alpha_t}\epsilon, t)\|^2$ 
7: until converged

```

Algorithm 2 Sampling

```

1:  $\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$ 
2: for  $t = T, \dots, 1$  do
3:    $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$  if  $t > 1$ , else  $\mathbf{z} = 0$ 
4:    $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left( \mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \alpha_t}} \epsilon_{\theta}(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$ 
5: end for
6: return  $\mathbf{x}_0$ 

```

Figure 2.24: **Pseudocode for training and sampling in DDPMs.** Algorithms from (Ho *et al.*, 2020b).

Despite their impressive performance, diffusion models can be computationally expensive, requiring hundreds or thousands of steps to sample. To address this, recent methods such as Latent Diffusion Models (LDMs) (Rombach *et al.*, 2022) apply diffusion in a learned latent space obtained via an autoencoder. This reduces the dimensionality of the data the diffusion model operates on, significantly improving both memory efficiency and sample quality.

Classifier-Free Guidance

Classifier-free guidance (CFG) (Ho and Salimans, 2022) is a technique for improving conditional diffusion sampling without using an external classifier. The model is trained to predict both conditional and unconditional scores by randomly dropping the conditioning variable $\mathbf{y}$ during training. At inference, the two predictions are combined:

$$\epsilon_{\text{CFG}}(\mathbf{x}_t, t) = (1 + w) \epsilon_{\theta}(\mathbf{x}_t, t \mid \mathbf{y}) - w \epsilon_{\theta}(\mathbf{x}_t, t \mid \emptyset) \quad (2.23)$$

where $\emptyset$ is the null condition and $w \geq 0$ controls adherence to the condition $\mathbf{y}$: larger w increases alignment to the condition but may reduce diversity. CFG is widely used in text-to-image and audio-visual diffusion models for its simplicity and effectiveness.

As with GANs, diffusion models are commonly employed in conditional settings, enabling controllable generation conditioned on inputs such as audio, pose, or identity features. They are increasingly applied to audio-visual domains, where temporally coherent synthesis and high-

quality generation are essential. For example, denoising diffusion models have been used to generate lip-synchronized video frames from speech embeddings (Bigata *et al.*, 2025a,b; Mittal and al., 2023; Li *et al.*, 2025) or to synthesize speech from visual inputs (Popov *et al.*, 2021). Unlike GANs, diffusion models are less prone to mode collapse and offer more stable training and higher quality samples, at the cost of longer inference time.

2.5.4 Flow Matching Models

Flow matching models are a class of generative models that learn to transform a simple base distribution (such as a standard Gaussian) into a complex target distribution through continuous-time dynamics (Lipman *et al.*, 2022). Unlike traditional normalizing flows (Rezende and Mohamed, 2015; Dinh *et al.*, 2017), which require invertible architectures and tractable Jacobian determinants, flow matching estimates a time-dependent velocity field that defines how samples evolve over time. This field is learned using a supervised objective derived from optimal transport, allowing the model to generate samples by solving an ordinary differential equation (ODE) with standard numerical solvers. Figure 2.25 illustrates that flow matching produces more structured and efficient sample trajectories than score-based alternatives.

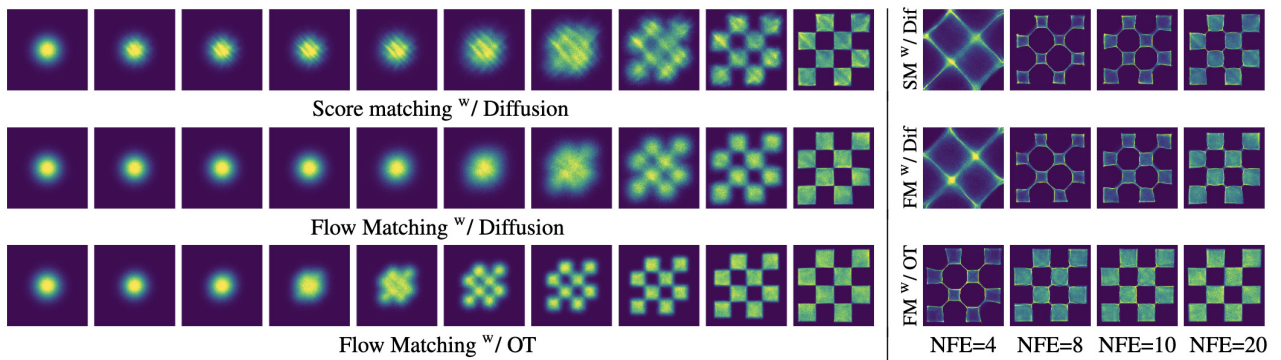


Figure 2.25: **A figure comparing trajectories of Continuous Normalizing Flows (CNFs).** (Left) CNF trajectories trained with three different objectives on 2D checkerboards. Optimal Transport (OT) based flows recover structure earlier, while diffusion-based objectives converge more slowly. (Right) Flow matching with OT produces higher-quality samples with fewer sampling steps. Figure from (Lipman *et al.*, 2022).

Training Process

Let $\mathbf{x}_0 \sim p_0$ be samples from a prior distribution (e.g., Gaussian), and $\mathbf{x}_1 \sim p_1$ be samples from the data distribution. Flow matching learns a velocity field $\mathbf{u}_\theta(\mathbf{x}, t)$ such that the solution to the ODE:

$$\frac{d\mathbf{x}_t}{dt} = \mathbf{u}_\theta(\mathbf{x}_t, t) \quad (2.24)$$

maps $\mathbf{x}_0$ to $\mathbf{x}_1$ as t moves from 0 to 1.

Flow matching avoids explicitly simulating trajectories during training by supervising the velocity field at intermediate interpolated points, with the path often chosen to be a simple linear interpolation:

$$\mathbf{x}_t = (1 - t)\mathbf{x}_0 + t\mathbf{x}_1, \quad t \sim \mathcal{U}[0, 1] \quad (2.25)$$

with a target velocity defined as its derivative with respect to t :

$$\mathbf{u}_{target}(\mathbf{x}_t, t) = \mathbf{x}_1 - \mathbf{x}_0 \quad (2.26)$$

The model is trained by minimizing the expected squared error between predicted and target velocities, essentially learning to model the probability path between p_0 to p_1 via a continuous flow:

$$\mathcal{L}_{FM} = \mathbb{E}_{\mathbf{x}_0, \mathbf{x}_1, t} [\|\mathbf{u}_\theta(\mathbf{x}_t, t) - \mathbf{u}_{target}(\mathbf{x}_t, t)\|^2] \quad (2.27)$$

This formulation allows end-to-end training of a network by iterating through the training dataset and sampling random noise $\mathbf{x}_0$, samples from our dataset $\mathbf{x}_1$, computing the interpolation $\mathbf{x}_t$ and the target vector flow $\mathbf{u}_{target}$.

Sampling Process

Once the velocity field $\mathbf{u}_\theta(\mathbf{x}, t)$ has been learned (Equation 2.24), new samples are generated by integrating it forward in time from an initial $t = 0$, $\mathbf{x}_0 \sim p_0$ to $t = 1$, $\mathbf{x}_1 \sim p_1$. In practice, this is done using a numerical ODE solver (e.g. Euler), which updates the sample in small steps. The process is conceptually similar to the reverse denoising procedure in diffusion models (Section 2.5.3), but here the evolution is deterministic and does not involve stochastic noise injection.

$$\mathbf{x}_{t+\Delta t} \approx \mathbf{x}_t + \Delta t \cdot \mathbf{u}_\theta(\mathbf{x}_t, t) \quad (2.28)$$

for a chosen step size Δt . Algorithm 2.26 shows the full sampling process. While this process is iterative, it is deterministic and generally faster than diffusion sampling due to the reduced number of steps and absence of stochastic noise injection. Classifier-free guidance (Section 2.5.3) can also be applied in flow matching models in a very similar way as diffusion.

Algorithm 3 Sampling

- 1: Sample $\mathbf{x}_0 \sim p_0$ // e.g., $\mathcal{N}(\mathbf{0}, \mathbf{I})$
 - 2: Set integration steps $t_0 = 0 < t_1 < \dots < t_N = 1$
 - 3: **for** $i = 0$ to $N - 1$ **do**
 - 4: Compute velocity $\mathbf{v} \leftarrow \mathbf{u}_\theta(\mathbf{x}_{t_i}, t_i)$
 - 5: Update state $\mathbf{x}_{t_{i+1}} \leftarrow \mathbf{x}_{t_i} + (t_{i+1} - t_i) \cdot \mathbf{v}$
 - 6: **end for**
 - 7: **return** $\mathbf{x}_{t_N}$ as generated sample
-

Figure 2.26: **Pseudocode for sampling in flow matching.**

While most generative modeling in audio-visual domains has focused on diffusion models, flow matching is increasingly being applied for its sampling efficiency and stable dynamics. Similar to diffusion-based methods, flow matching can be conditioned on acoustic features, identity embeddings, or temporal priors to generate coherent face sequences or cross-modal outputs (Cho *et al.*, 2023; Wang, Liu, *et al.*, 2024; Du *et al.*, 2025).

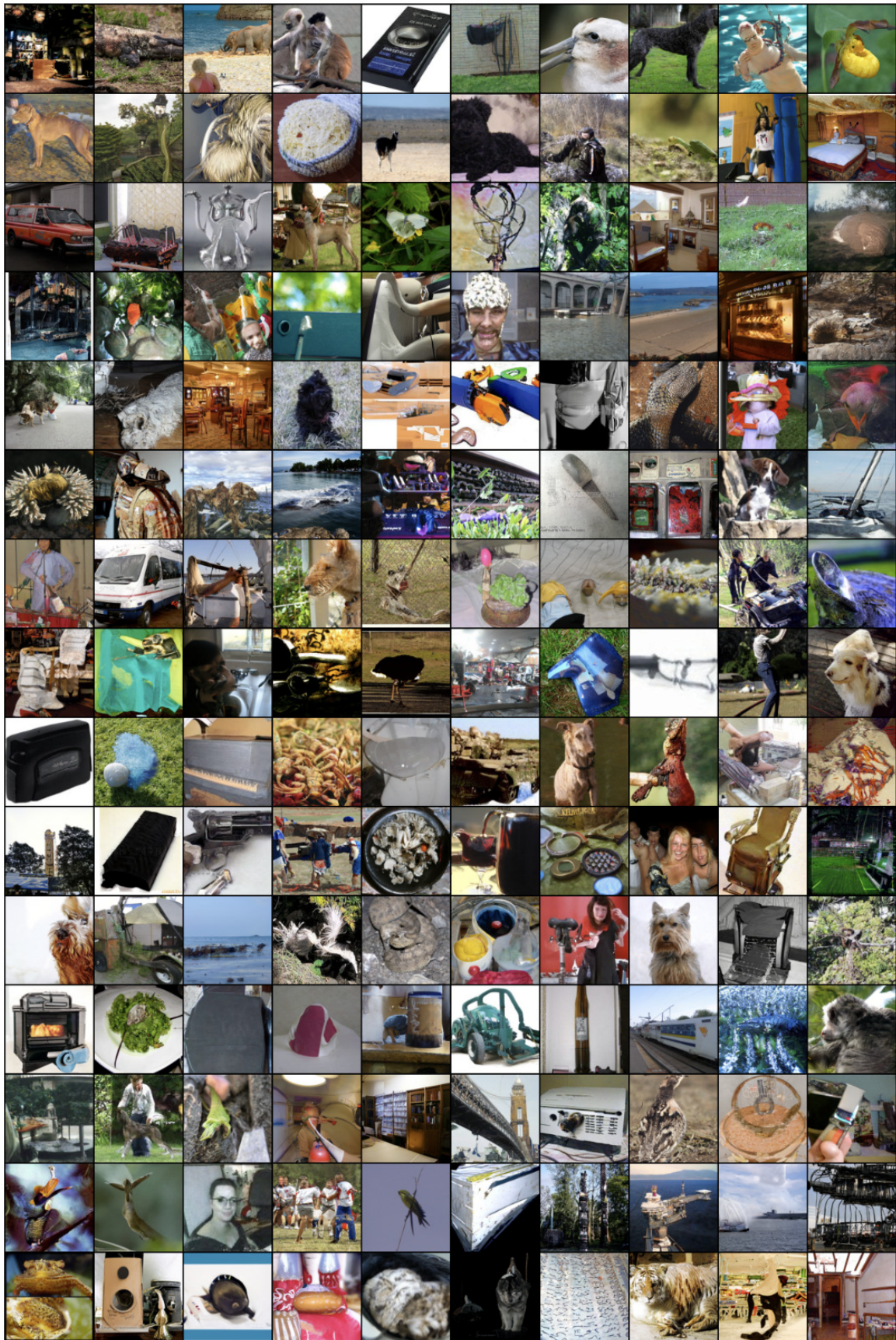


Figure 2.27: **Some generated samples with flow matching with optimal transport.** Samples are of resolution 128×128 and generated unconditionally from ImageNet (Deng *et al.*, 2009). Figure from (Lipman *et al.*, 2022).

2.6 Optimisers

Optimisers are algorithms that update the parameters of a neural network during training in order to minimise a loss function. Below, we give an overview of the most fundamental methods. In our works, we employ AdamW (Loshchilov and Hutter, 2019) as the primary optimiser, combined with learning rate schedules.

2.6.1 Stochastic Gradient Descent (SGD)

Stochastic gradient descent (SGD) is one of the foundational optimisation methods for neural networks (Rumelhart *et al.*, 1986). At each iteration t , model parameters $\boldsymbol{\theta}_t$ are updated in the opposite direction of the gradient $\nabla_{\boldsymbol{\theta}}$ of the loss $\mathcal{L}$ computed over a mini-batch of samples:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_t), \quad (2.29)$$

where η is the learning rate. SGD is simple and effective but often requires careful tuning of η and converges slowly in practice. Momentum variants extend SGD by accumulating a moving average of past gradients, helping to smooth updates and accelerate convergence.

2.6.2 Adam

Adam (Adaptive Moment Estimation) (Kingma and Ba, 2014) extends momentum by computing adaptive learning rates for each parameter. For gradients $\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_t)$, Adam updates the model weights as follows:

$$\mathbf{m}_t = \beta_1 \mathbf{m}_{t-1} + (1 - \beta_1) \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_t) \quad (2.30)$$

$$\mathbf{v}_t = \beta_2 \mathbf{v}_{t-1} + (1 - \beta_2) (\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_t))^2 \quad (2.31)$$

$$\hat{\mathbf{m}}_t = \frac{\mathbf{m}_t}{1 - \beta_1^t}, \quad \hat{\mathbf{v}}_t = \frac{\mathbf{v}_t}{1 - \beta_2^t} \quad (2.32)$$

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \frac{\hat{\mathbf{m}}_t}{\sqrt{\hat{\mathbf{v}}_t} + \epsilon}, \quad (2.33)$$

where $\mathbf{m}_t$ and $\mathbf{v}_t$ are the first and second moment estimates and $\hat{\mathbf{m}}_t$ and $\hat{\mathbf{v}}_t$ are the bias correction terms for mean and variance respectively, $\beta_1, \beta_2 \in [0, 1)$ control the decay rates, t is the iteration number and ϵ is a small constant for numerical stability. Adam is widely adopted for its fast convergence and robustness to hyperparameter choices.

2.6.3 AdamW

AdamW (Loshchilov and Hutter, 2019) modifies Adam by decoupling weight decay from the gradient update. In Adam, weight decay is implemented as an ℓ_2 regularisation term added to the loss function, which can interfere with the adaptive scaling of learning rates. Instead, AdamW applies weight decay directly in the parameter update step:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \left(\frac{\hat{\mathbf{m}}_t}{\sqrt{\hat{\mathbf{v}}_t} + \epsilon} + \lambda \boldsymbol{\theta}_t \right), \quad (2.34)$$

where λ is the weight decay coefficient. This simple change improves training stability and generalisation, and has become increasingly used in large-scale deep learning.

2.6.4 Learning Rate Schedules

In addition to the choice of optimiser, the learning rate schedule critically influences training dynamics, improving both convergence and generalization in deep neural networks. A common strategy is to employ a warmup phase, where the learning rate is gradually increased from zero to its target value over the first T_{warmup} iterations, reducing the risk of divergence in the early stages of training. After warmup, the learning rate can remain constant or be decayed according to a schedule. In our speech recognition work, we adopt cosine decay (Loshchilov and Hutter, 2017), where the learning rate follows a cosine function:

$$\eta_t = \eta_{\min} + \frac{1}{2}(\eta_{\max} - \eta_{\min}) \left(1 + \cos\left(\frac{t}{T}\pi\right) \right), \quad (2.35)$$

with $\eta_{\min}$ and $\eta_{\max}$ are the minimum and maximum learning rates, t is the current step, and T the total number of decay steps. In our lip synchronization work, we instead use the 1-cycle learning rate policy (OneCycleLR) (Smith and Topin, 2017). This schedule first increases the learning rate from a lower initial value to a peak value, and then anneals it to a final value that is typically lower than the starting point.

2.7 Loss Functions

Loss functions define the optimisation objectives for deep learning models by quantifying the discrepancy between a model’s predictions and the target outputs. Training then proceeds by adjusting the model parameters to minimise this discrepancy. In this section, we describe the loss functions used in our work, grouped into two categories – speech recognition and lip synchronization – according to the task they were applied to.

2.7.1 Speech Recognition

Cross-Entropy Loss

Cross-entropy is the standard loss for multiclass classification. Given unnormalized logits $\mathbf{z} \in \mathbb{R}^M$ for M classes, the softmax function maps them into a probability distribution:

$$\hat{y}_i = \frac{\exp(z_i)}{\sum_{j=1}^M \exp(z_j)}. \quad (2.36)$$

With ground truth label vector $\mathbf{y} \in \{0, 1\}^M$, the cross-entropy loss is

$$\mathcal{L}_{\text{CE}} = - \sum_{i=1}^M y_i \log \hat{y}_i. \quad (2.37)$$

This objective is minimised when the predicted probability of the correct class approaches 1, and 0 for all other classes.

Negative Cosine Similarity Loss

Cosine similarity encourages a predicted vector $\hat{\mathbf{y}} \in \mathbb{R}^d$ and a target vector $\mathbf{y} \in \mathbb{R}^d$ (or any two vectors) to align in direction, regardless of their magnitudes. The similarity is defined as:

$$\mathcal{L}_{\text{cos}} = \frac{\mathbf{y}^\top \hat{\mathbf{y}}}{\|\mathbf{y}\| \|\hat{\mathbf{y}}\|} \quad (2.38)$$

This value lies in the range $[-1, 1]$, where 1 indicates perfect alignment, 0 indicates orthogonality, and -1 indicates that the vectors point in opposite directions. To use this measure as a loss to minimise in deep learning, we negate it, yielding the negative cosine similarity loss:

$$\mathcal{L}_{\text{negcos}} = -\mathcal{L}_{\text{cos}} \quad (2.39)$$

Minimising this objective encourages $\hat{\mathbf{y}}$ to align directionally with $\mathbf{y}$. Unlike the ℓ_2 distance, cosine similarity is scale-invariant.

Connectionist Temporal Classification Loss

Connectionist Temporal Classification (CTC) (Graves *et al.*, 2006) is a loss function tailored for sequence-to-sequence tasks where the input and label sequences differ in length and the alignment between them is unknown. Let the input sequence be $\mathbf{x}_{1:T}$ of length T and the target label sequence be $\mathbf{y}_{1:U}$ of length $U \leq T$. CTC introduces a blank symbol ($\emptyset$) and considers all possible length- T alignment sequences (or paths) $\pi = (\pi_1, \dots, \pi_T)$, where each π_t is either a label or the blank.

Each path π is mapped to a collapsed label sequence by removing blanks and merging repeated symbols. The CTC model defines the conditional probability of a label sequence given the input by marginalizing over all paths that collapse to it:

$$p(\mathbf{y} \mid \mathbf{x}) = \sum_{\pi \in B^{-1}(\mathbf{y})} \prod_{t=1}^T p_t(\pi_t \mid \mathbf{x}), \quad (2.40)$$

where $\mathcal{B}^{-1}(\mathbf{y})$ denotes the set of all possible paths of length T that collapse to $\mathbf{y}$, and $p_t(\pi_t \mid \mathbf{x})$ is the probability of observing the symbol π_t at time t .

The CTC loss is the negative log-likelihood of the correct label sequence under this model:

$$\mathcal{L}_{\text{CTC}} = -\log p(\mathbf{y} \mid \mathbf{x}). \quad (2.41)$$

During training, dynamic programming (a forward-backward algorithm) is used to efficiently compute the sum over all valid alignments. CTC enforces a monotonic alignment (labels must appear in order) but allows flexible timing, enabling repeated or blank frames between labels. This alignment flexibility makes CTC particularly effective for end-to-end sequence learning tasks such as speech recognition.

Negative Log-Likelihood (Attention-based) Loss

In encoder-decoder models with attention, the output sequence $\mathbf{y}_{1:T}$ is generated auto-regressively given the previous sequence predictions and encoder output $\mathbf{x}$: at each step t , the model predicts $p(y_t \mid y_{1:t-1}, \mathbf{x})$. The loss is the negative log-likelihood of this probability:

$$\mathcal{L}_{\text{Att}} = -\sum_{t=1}^T \log p(y_t \mid y_{1:t-1}, \mathbf{x}) \quad (2.42)$$

It encourages the decoder to maximize the probability of the correct sequence while leveraging the context computed through attention. During training, teacher forcing (Williams and Zipser, 1989) is commonly used, where the decoder conditions on the available ground-truth $y_{1:t-1}$ instead of its own past predictions, allowing the loss to be computed in parallel across all time steps.

Joint CTC-Attention Loss

CTC and attention-based objectives are complementary: CTC provides alignment constraints between input frames and target symbols, while attention provides flexible contextual modeling

of sequence dependencies. They can be combined into a multi-task objective of the form:

$$\mathcal{L}_{\text{MTL}} = \lambda \mathcal{L}_{\text{CTC}} + (1 - \lambda) \mathcal{L}_{\text{Att}} \quad (2.43)$$

where $\lambda \in [0, 1]$ controls the trade-off between the two terms (Kim *et al.*, 2017). Optimizing this joint loss encourages models to benefit from the monotonic alignment structure imposed by CTC while retaining the sequence modeling ability of attention.

Decoding. While the losses above define training objectives, during inference the model must decode sequences from probability distributions over symbols. A common strategy we also employ is beam search (Freitag and Al-Onaizan, 2017), which maintains the top- k candidate sequences at each decoding step and expands them iteratively. The final output is selected as the prediction with the highest accumulated log-probability. Beam search strikes a balance between greedy decoding and exhaustive search, yielding more accurate outputs without excessive computational cost.

2.7.2 Lip Synchronization

Mean Absolute Error (MAE)

The Mean Absolute Error (MAE) loss, also known as ℓ_1 loss, is a loss function used in machine learning, particularly for regression and image/video generation tasks. It measures the average of the absolute differences between the predicted values $\hat{\mathbf{y}}$ of a model and the ground truth values $\mathbf{y}$. The loss is defined as:

$$\mathcal{L}_{\text{MAE}} = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i| \quad (2.44)$$

where n is the total number of data points and $|\cdot|$ is the absolute value function. It is less sensitive to outliers than the MSE (or ℓ_2) loss since it does not use squaring.

Mean Squared Error (MSE)

The Mean Squared Error (MSE) loss, also known as ℓ_2 loss, is a fundamental metric for evaluating models in regression problems. It measures the average of the squared difference between the predicted values $\hat{\mathbf{y}}$ and the ground truth values $\mathbf{y}$. The loss is defined as:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (2.45)$$

where n is the total number of data points. Squaring the errors means that larger deviations lead to a higher penalty, rendering MSE more susceptible to outliers compared to MAE.

Perceptual/VGG Loss

Perceptual loss, often called *VGG loss* as it uses a pre-trained VGG network (Simonyan and Zisserman, 2014), is a loss function for image/video generation tasks. Unlike pixel-wise losses, it measures high-level perceptual differences between a generated image $\hat{\mathbf{x}}$ and its ground truth $\mathbf{x}$ (Johnson *et al.*, 2016).

It works by passing both images through a pre-trained CNN (typically VGG-16/19). The loss is then the ℓ_1 or ℓ_2 distance between their feature representations extracted from intermediate layers. The formula for an ℓ_2 -based perceptual loss is:

$$\mathcal{L}_{\text{Perceptual}} = \sum_j \frac{1}{C_j H_j W_j} \|\phi_j(\mathbf{x}) - \phi_j(\hat{\mathbf{x}})\|_2^2 \quad (2.46)$$

where $\phi_j(\mathbf{x})$ is the feature map from layer j , and C_j, H_j, W_j are its dimensions. This approach captures more semantically meaningful information, leading to perceptually more realistic generated images and mitigating the blurriness often seen with pixel-wise losses.

2.8 Evaluation Metrics

Evaluation metrics provide a quantitative way to assess how well a model performs on its intended task. While loss functions guide the optimisation during training, evaluation metrics are used after training to compare models, assess generalisation, and measure progress towards specific goals. Similarly to before, we divide the metrics into two categories: speech recognition and generative modeling.

2.8.1 Speech Recognition

Word Error Rate (WER)

Word Error Rate (WER) is a common metric for evaluating automatic speech recognition (ASR) systems. It measures the edit distance between a predicted word sequence and the ground truth sequence, normalised by the number of words in the reference:

$$\text{WER} = \frac{S + D + I}{N} \quad (2.47)$$

where S is the number of substitutions, D the number of deletions, I the number of insertions, and N the total number of words in the ground truth sequence. Lower WER indicates better performance. WER captures intelligibility at the word level, but can be sensitive to minor transcription variations such as synonyms or morphology.

Character Error Rate (CER)

Character Error Rate (CER) is analogous to WER but computed at the character level:

$$\text{CER} = \frac{S + D + I}{N} \quad (2.48)$$

where here S, D, I are computed over characters rather than words, and N is the number of characters in the ground truth. CER is particularly relevant in languages with large vocabularies

not separated by spaces like Mandarin or in low-resource settings where character-level modeling is more robust over word-level modeling.

2.8.2 Lip Synchronization

Fréchet Inception Distance (FID)

The Fréchet Inception Distance (FID) (Heusel *et al.*, 2017) evaluates the distributional similarity between generated and real data. Features are extracted from a pretrained Inception network for both real samples $\mathbf{x}$ and generated samples $\hat{\mathbf{x}}$, yielding Gaussian distributions with means $\boldsymbol{\mu}_r, \boldsymbol{\mu}_g$ and covariances $\boldsymbol{\Sigma}_r, \boldsymbol{\Sigma}_g$. FID is then computed as:

$$\text{FID} = \|\boldsymbol{\mu}_r - \boldsymbol{\mu}_g\|_2^2 + \text{Tr}(\boldsymbol{\Sigma}_r + \boldsymbol{\Sigma}_g - 2(\boldsymbol{\Sigma}_r \boldsymbol{\Sigma}_g)^{1/2}) \quad (2.49)$$

where $\text{Tr}(\cdot)$ denotes the matrix trace operator, i.e. the sum of the diagonal elements. Lower FID indicates that the generated distribution is closer to the real data distribution. It correlates reasonably well with visual quality, though it is sensitive to dataset size and choice of feature extractor.

Fréchet Video Distance (FVD)

Fréchet Video Distance (FVD) (Unterthiner *et al.*, 2019a) extends FID to video by comparing spatiotemporal feature distributions. It uses embeddings from an Inflated 3D ConvNet (I3D) pretrained on action recognition. Given real and generated video distributions with means $\boldsymbol{\mu}_r, \boldsymbol{\mu}_g$ and covariances $\boldsymbol{\Sigma}_r, \boldsymbol{\Sigma}_g$, FVD is defined analogously to FID using Eq. 2.49. Unlike frame-based metrics, FVD accounts for temporal dynamics and consistency across frames, making it better suited for evaluating generative video models.

HyperIQA Score

The HyperIQA metric (Su *et al.*, 2020) is a no-reference image quality assessment method based on a self-adaptive *hyper network*. An input image $\mathbf{x}$ is first processed by a feature extractor f to obtain deep visual features. These features are passed to a hyper network h , which predicts the parameters $\boldsymbol{\theta}$ of a content-dependent quality prediction head. The final image-quality score q is then produced by applying a prediction network $\phi(\cdot; \boldsymbol{\theta})$ to the same features:

$$\boldsymbol{\theta} = h(f(\mathbf{x})), \quad q = \phi(f(\mathbf{x}); \boldsymbol{\theta}). \quad (2.50)$$

This design allows the model to adapt its quality estimation to the content and distortion characteristics of each input image without requiring a reference image.

VBench Metrics

VBench (Huang *et al.*, 2023) is an open-source benchmark suite designed to evaluate video generative models across a wide range of dimensions, such as subject consistency, visual quality, and motion smoothness. For our lip synchronization task, we use only the subset of VBench metrics that are directly relevant to audio-driven facial generation.

LipScore

LipScore (Bigata *et al.*, 2025a) is a metric designed to assess lipreading visual quality. It evaluates how well the generated mouth movements align compared to the ground-truth by computing the cosine similarity between their embeddings, extracted from the final layer of the Auto-AVSR lipreader (Ma *et al.*, 2023). This approach addresses limitations of earlier methods such as SyncNet (Prajwal *et al.*, 2020a), which have been shown to correlate weakly with true lip-sync quality.

Identity Preservation

Identity preservation quantifies how well a generative model maintains the distinctive characteristics of a person. These metrics are typically derived from face-similarity scores obtained using face recognition models. In our work, we measure identity preservation as the cosine similarity between predicted and ground-truth face embeddings, computed per frame using FaceNet-512 (Schroff *et al.*, 2015; Serengil and Ozpinar, 2024).

Structural Similarity Index (SSIM)

The Structural Similarity Index (SSIM) is a perceptual metric that measures the similarity between two images by comparing their luminance, contrast, and structural information. Unlike pixel-wise measures such as MSE or PSNR, SSIM correlates more closely with human perceptual judgments of image quality. It is defined as:

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}, \quad (2.51)$$

where μ_x and μ_y denote local means of images x and y , σ_x^2 and σ_y^2 their variances, σ_{xy} their covariance, and C_1, C_2 are small constants to ensure numerical stability.

Peak Signal-to-Noise Ratio (PSNR)

Peak Signal-to-Noise Ratio (PSNR) is derived from the MSE and measures the ratio between the maximum possible signal value and the error power:

$$\text{PSNR}(x, y) = 10 \cdot \log_{10} \left(\frac{L^2}{\text{MSE}(x, y)} \right) \quad (2.52)$$

where L is the maximum possible pixel value (e.g. 255 for 8-bit images). Higher PSNR values indicate better reconstruction fidelity. PSNR is easy to compute and interpret, but like MSE, it does not always align with human perception of realism.

Learned Perceptual Image Patch Similarity (LPIPS)

LPIPS (Zhang *et al.*, 2018) measures perceptual similarity between images by comparing deep feature activations. Given two images $\mathbf{x}$ and $\hat{\mathbf{x}}$, they are passed through a pretrained network (e.g. AlexNet (Krizhevsky *et al.*, 2012) or VGG), and the difference between channel-normalised feature maps $\phi_l(\cdot)$ across multiple layers l is aggregated:

$$\text{LPIPS}(\mathbf{x}, \hat{\mathbf{x}}) = \sum_l \frac{1}{H_l W_l} \sum_{h,w} \|\mathbf{w}_l \odot (\phi_l(\mathbf{x})_{hw} - \phi_l(\hat{\mathbf{x}})_{hw})\|_2^2 \quad (2.53)$$

where H_l, W_l are the height and width of the feature map, h, w are the spatial coordinates, $\mathbf{w}_l$ is a learnable layer-wise vector, and $\|\cdot\|_2$ is the ℓ_2 norm (Euclidean distance). Lower LPIPS indicates greater perceptual similarity, aligning more closely with human judgments than PSNR or MSE.

CHAPTER 3

LEARNING CROSS-LINGUAL VISUAL SPEECH REPRESENTATIONS

Contents

3.1	Introduction	63
3.2	Datasets	64
3.3	Method	65
3.3.1	Pre-training	65
3.3.2	Fine-tuning	66
3.4	Experiments	67
3.4.1	Pre-processing	67
3.4.2	Data augmentation	67
3.4.3	Masking	67
3.4.4	Training settings	67
3.4.5	Fine-tuning	68

3.5	Results	68
3.5.1	Multilingual pre-training surpasses monolingual	68
3.5.2	Multilingual pre-training surpasses English-only pre-training with the same number of hours	69
3.5.3	Monolingual pre-training surpasses multilingual with same hours	69
3.5.4	Related language pre-training performs better	71
3.5.5	Representations on unseen languages	71
3.6	Limitations	72
3.7	Conclusion	72

3.1 Introduction

Cross-lingual learning uses data from different languages to create more robust and accurate models. Relevant supervised approaches (Ma *et al.*, 2022; Burget *et al.*, 2010; Heigold *et al.*, 2013; Kannan *et al.*, 2019) have delivered promising results, showing that increasingly larger models and training sets substantially improve performance. However, training such powerful models requires large amounts of labelled data, which can be time-consuming and expensive to acquire, hindering scalability.

Recently, several self-supervised learning approaches have been proposed which leverage large amounts of unlabelled data to learn better representations (He *et al.*, 2020; Grill *et al.*, 2020; Hsu *et al.*, 2021; Devlin *et al.*, 2019; Haliassos *et al.*, 2022; Baevski *et al.*, 2020a, 2019) and then fine-tune on the downstream tasks such as speech recognition, object detection, etc. Semi-supervised works which focus on various accents or dialects have also been released (Hu *et al.*, 2021; Li *et al.*, 2018). Some self-supervised approaches have also been proposed for learning visual speech representations, achieving promising results (Shi *et al.*, 2022; Haliassos *et al.*, 2023; Ma *et al.*, 2021b). However, these works tend to use English-only data both for pre-training and fine-tuning, and it is not clear how they perform when non-English labelled or

unlabelled data is used. A couple of self-supervised works (Conneau *et al.*, 2021; Babu *et al.*, 2022) learn cross-lingual speech representations, but focus on improving the recognition of acoustic speech using multilingual models and do not compare monolingual models of the target language with the same pre-training dataset size. To the best of our knowledge, there is no existing work focusing on learning cross-lingual visual speech representations.

In this chapter, we aim to leverage a self-supervised method to learn visual representations for multiple languages. We use the RAVEn framework (Haliassos *et al.*, 2023), where visual and audio encoders are pre-trained on unlabelled data via cross- and within-modal losses. The pre-trained visual backbone is then fine-tuned for the recognition of visual speech.

We make the following findings: (1) Pre-training on multiple languages leads to a lower character error rate (CER) compared to monolingual pre-training when more data is used, but to a higher CER when keeping the number of pre-training hours fixed; (2) multi-lingual pre-training outperforms English-only pre-training with the same number of data hours, showing that dataset size is not solely responsible for the improvement and that the information from other languages likely plays a significant role; (3) pre-training and fine-tuning on similar languages achieves better performance compared to less similar ones, signifying that the choice in languages is important; (4) fine-tuning on languages unseen during pre-training reaches competitive results compared to when including the language in the pre-training set.

3.2 Datasets

We use four large-scale publicly-available audio-visual datasets for our study. For pre-training, we use the multilingual AVSpeech dataset (Ephrat *et al.*, 2018), while for fine-tuning we use LRS3 (Afouras *et al.*, 2018b) for English (en), Multilingual TEDx (Salesky *et al.*, 2021) for Spanish (es), Italian (it), Portuguese (pt) and French (fr), and Chinese Mandarin Lip Reading (CMLR) (Zhao *et al.*, 2019) for Mandarin (zh). Details on the datasets can be found in Section 2.1.1.

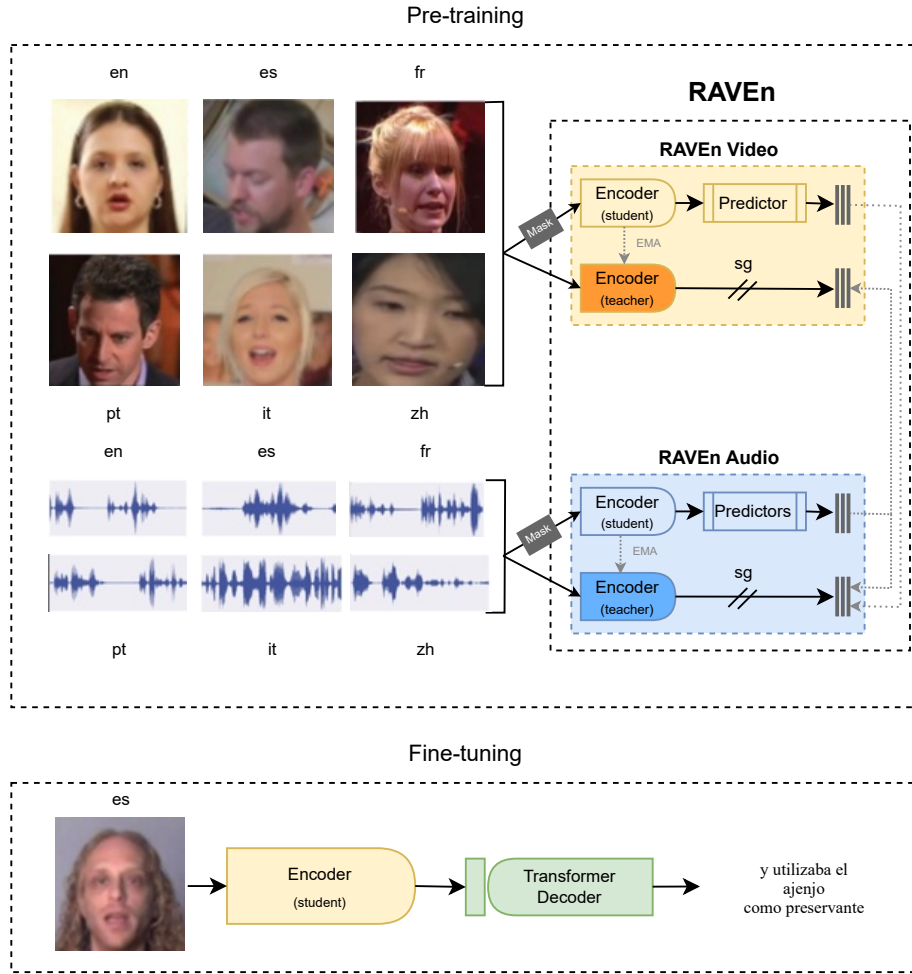


Figure 3.1: **Pre-training and fine-tuning pipelines.** We use RAVEn (Haliassos *et al.*, 2023) to pre-train models using audio and video from multiple languages. Stop-gradient and Exponential Moving Average are denoted by *sg* and *EMA*, respectively. We then initialise the visual-only encoder using the weights from the pre-training stage and fine-tune the entire network using the language-specific dataset.

3.3 Method

3.3.1 Pre-training

We use the audio-visual self-supervised RAVEn (Haliassos *et al.*, 2023) method for our study, as shown in Fig. 3.1. It consists of two pairs of student-teacher models, one for video and one for audio. The students intake masked inputs and predict targets generated by momentum-based teachers (Grill *et al.*, 2020), which are given unmasked inputs. The video encoder consists of a

ResNet-18 (He *et al.*, 2016) with a 3D stem (Stafylakis and Tzimiropoulos, 2017), followed by a Transformer encoder (Vaswani *et al.*, 2017) with attention dimension 512, 8 heads, 2048 hidden units, and 12 layers. The audio encoder comprises a 1D ResNet-18 (Petridis *et al.*, 2018), with the output piped into a Transformer with the same settings as for the video encoder. The teachers have equivalent architectures to their student counterparts, apart from the students also containing additional 2-block Transformer predictors with attention dimension 512, 8 heads and 2048 linear units, which use mask tokens corresponding to the masked input. The predictors prevent representation collapse (Grill *et al.*, 2020).

The audio student predicts both the video and audio targets, while the video student predicts the audio targets only. The loss between the predictions and targets is the cosine similarity. The students are optimised with a usual gradient-based method using the following loss:

$$\mathcal{L} = \text{sim}(p_s^a, \text{sg}(e_t^v)) + \text{sim}(p_s^v, \text{sg}(e_t^a)) + \text{sim}(p_s^a, \text{sg}(e_t^a)), \quad (3.1)$$

where sim is the cosine similarity function, sg the stop-gradient, p_s^a and p_s^v the student predictions, and e_t^a and e_t^v the teacher encodings for audio and video respectively. The teacher models are updated using an exponential moving average (EMA) of the student weights using the following equation:

$$\xi^m \leftarrow \tau \xi^m + (1 - \tau) \theta^m, \quad (3.2)$$

for momentum parameter τ , modality $m \in \{v, a\}$, student weights θ and teacher weights ξ , with τ starting at 0.999 and following a cosine schedule to 1.

3.3.2 Fine-tuning

Following (Haliassos *et al.*, 2023), we fine-tune the pre-trained visual student encoder to perform visual speech recognition by attaching a linear layer and a Transformer decoder with attention dimension 256, 4 heads, 6 layers and 2048 linear units. We apply a joint CTC / attention loss (Watanabe *et al.*, 2017). The beam size and CTC weight are fixed to 40 and

0.1 respectively, as in (Ma *et al.*, 2021c). For labels, we use a character set for Mandarin and subword units (Kudo and Richardson, 2018) of vocabulary size 1000 for all other languages.

3.4 Experiments

3.4.1 Pre-processing

Dataset pre-processing follows (Ma *et al.*, 2022), and is described in more detail in Section 2.2.2. Audio pre-processing is described in Section 2.2.1.

3.4.2 Data augmentation

During training, we randomly crop the input video to a size of 88×88 and perform horizontal flipping with probability 0.5 (Ma *et al.*, 2022). The transformations are applied throughout the temporal dimension of the clip consistently. No augmentations are performed for the audio.

3.4.3 Masking

Our masking strategy in pre-training follows (Haliassos *et al.*, 2023): we perform random sampling on the video frames with probability 0.2 to select any as the start mask index, from where we zero out the next three frames. We apply a similar mask to the audio clip with a scaling factor of 640 to account for the sampling size difference compared to video.

3.4.4 Training settings

We pre-train for 150 epochs with a peak learning rate of $3e-3$, a linear warm-up for the first 40 epochs, and a cosine decay for the remaining epochs. The optimiser used is AdamW (Loshchilov and Hutter, 2019) with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and weight decay 0.04.

We fine-tune for 50 epochs, with a 20-epoch warm-up and a cosine learning rate decay. We use a learning rate of 1e-3 for the encoder with layer-wise decay with parameter 0.5 (Clark *et al.*, 2020), and a learning rate of 5e-3 for the decoder. The optimiser is AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.98$ and weight decay 0.1.

3.4.5 Fine-tuning

We fine-tune on a single language at a time. A subset of LRS3 (“trainval” partition) with a total of 30 hours is used to fine-tune in English. Multilingual TEDx is used to fine-tune in Spanish (62 h), French (72 h), Portuguese (73 h) and Italian (40 h). CMLR is used to fine-tune in Mandarin (60.5 h). We measure performance using Character Error Rate (CER) instead of Word Error Rate (WER), since Mandarin characters are not separated by spaces and to maintain consistency with the other languages.

3.5 Results

3.5.1 Multilingual pre-training surpasses monolingual

We pre-train (1) six monolingual models with either English, French, Italian, Spanish, Portuguese, or Mandarin (“*Monolingual*”); (2) one multilingual model with all six languages combined (“*Multilingual*”); and (3) one with all languages combined but with the same number of hours as for the English-only dataset (“*Multilingual RE*”), since AVSpeech contains more English-speaking videos than other languages¹. Table 3.1 shows that for every language except English, multilingual pre-training significantly outperforms monolingual. We hypothesise that the extra data helps lower-resource languages by enabling the model to learn structures of speech shared across different languages. For English, which contains plenty of pre-training data, adding more languages may cause the representations to become less English-specific, in turn harming downstream performance.

¹We combine all languages except English and then randomly add English samples until we reach 1 333 hours for the Reduced English (RE) model. This results in 542 hours of English and 791 hours of other languages.

Table 3.1: **Character Error Rate (CER) of visual-only models for different languages.** We pre-train models on either one language or six, and fine-tune on each language independently. English-only refers to the English monolingual model.

Model	Languages					
	en	it	pt	fr	es	zh
Pre-Trained Data (Hours)	1,333	68	337	117	204	65
Fine-Tuned Data (Hours)	30	40	73	72	62	60.5
English-Only	28.5	39.0	41.6	44.8	35.9	16.2
Monolingual	28.5	76.9	43.5	71.9	47.2	22.6
Multilingual	31.5	38.0	40.0	43.6	35.3	15.8
Multilingual RE	32.8	35.1	38.7	43.3	32.8	15.9
Multilingual MH	32.8	75.7	48.3	73.3	49.6	24.4

3.5.2 Multilingual pre-training surpasses English-only pre-training with the same number of hours

Seeing that multilingual models perform better when pre-trained with more data (Section 3.5.1), we investigate whether simply adding more data explains the improvement. We trained a monolingual “*English-only*” model and a “*Multilingual RE*” model both with 1333 hours, as explained in section 3.5.1. We fine-tune both models on all languages independently, and see that the “*English-only*” model outperforms the other monolingual models, signifying that the number of training hours does matter. However the “*Multilingual RE*” model outperforms the English-only model in every language except English, as seen in Table 3.1. This shows that the addition of more data in itself does not fully explain the performance boost: the information from various languages gives a further reduction in the error even when using the same dataset size.

3.5.3 Monolingual pre-training surpasses multilingual with same hours

We now compare multilingual and monolingual models, keeping the number of training hours constant, to disentangle the effects of extra data from the language variety (see Table 3.1). We train six multilingual models (“*Multilingual MH*”) (Matched Hours), each with equal parts of every language to add up to the same number of hours used for the monolingual model. For

Table 3.2: **CER on low-resource languages.** We pre-train six monolingual models with 30 hours of data randomly sampled from each language, and one multilingual model with the same hours of pre-training data by sampling 5 hours from each language randomly. We compare the results in the case where the languages are low-resource.

Model	Languages					
	en	it	pt	fr	es	zh
Pre-Trained Data (Hours)	30	30	30	30	30	30
Fine-Tuned Data (Hours)	30	40	73	72	62	60.5
Monolingual	79.8	79.1	75.1	75.7	82.6	97.4
Multilingual	84.4	82.4	77.8	77.5	90.0	99.7

Table 3.3: **CER comparison when fine-tuning Spanish, pre-training on Spanish with added languages.** We pre-train models on three different pairs of languages without the use of any labelled data, and then fine-tune the visual model using a total of 62 hours of labelled video clips in Spanish.

Model	Pre-Trained Data (Hours)	es
Spanish	204	47.2
Spanish + Mandarin	$204 + 65 = 269$	43.6
Spanish + French	$204 + 65 = 269$	42.7
Spanish + Portuguese	$204 + 65 = 269$	40.5

every language except Italian, the monolingual model surpasses the multilingual one. We also consider a low-resource scenario, in which we randomly sample 30 hours of each language and train six new monolingual models, as well as a multilingual one with 30 hours of data in total, sampling 5 hours from each language. Table 3.2 shows that for every language in this context, monolingual outperforms multilingual pre-training.

This leads us to conclude that having sufficient hours of training data in the target language is important and directly substituting with equivalent data hours from other languages will worsen performance. Nonetheless, assuming that the amount of data from the target language is held constant, adding additional data from other languages improves representation learning, as discussed in Section 3.5.1.

Table 3.4: **CER on unseen languages.** We pre-train two multilingual models: one excluding Portuguese and one excluding Spanish, using the same number of pre-training hours as the Multilingual RE model we are comparing to.

Model	pt	es
Fine-Tuned Data (Hours)	73	62
Multilingual RE	38.7	32.8
Multilingual – Unseen (pt)	41.5	-
Multilingual – Unseen (es)	-	34.0

3.5.4 Related language pre-training performs better

We hypothesise that, keeping the number of hours fixed, pre-training on languages more related to the target language will lead to better results than pre-training on dissimilar languages. To test this hypothesis, we pre-train three different models: Spanish (204 h) combined with either Portuguese (65 h), French (65 h)² or Mandarin (65 h), and fine-tune on Spanish, as shown in Table 3.3. By adding Mandarin for pre-training, we observe an absolute reduction of 3.6 % in CER. By replacing Mandarin data with French, a further improvement of 0.9 % can be observed. The CER is further reduced to 40.5 % with an absolute reduction of 2.2 % when using Portuguese instead. This is likely due to the fact that Portuguese is the most similar language to Spanish, which benefits the learning of visual speech representations for the language.

3.5.5 Representations on unseen languages

To investigate how well the learned visual speech representations perform on unseen languages, we train two multilingual models with all languages except either Spanish or Portuguese, totalling 1 333 hours of pre-training data³, the same as the “*Multilingual RE*” model. We fine-tune on the unseen language, Portuguese and Spanish respectively. Results are shown in Table 3.4. For comparison purposes, we use “*Multilingual RE*” model as our baseline. We show that the performance is competitive to the model for which the language was included in the data.

²We randomly sample the Portuguese and French datasets until we reach 65 hours.

³We add English data to reach 1 333 hours when removing each language.

3.6 Limitations

Our work focuses exclusively on the visual speech recognition downstream task, whereas audio cues play an important role in speech understanding, particularly across languages. Extending this approach to audio-visual speech recognition could therefore provide additional insights and presents a promising direction for future work. Furthermore, our results indicate that multilingual pre-training introduces a trade-off between learning more robust but generic representations and learning higher-performing, language-specific representations when the total amount of training data is fixed; this trade-off should be considered depending on data availability and application requirements. Finally, we observe that multilingual pre-training is more effective when using linguistically similar languages rather than arbitrary language combinations, an important consideration when designing cross-lingual training setups.

3.7 Conclusion

In this chapter, we investigated the impact of multilingual pre-training on visual speech recognition using a recently proposed self-supervised framework. We show that, on average, multilingual pre-training surpasses monolingual when pre-trained with more data by 23.8% and 25.0% in relative terms when comparing “Monolingual” with the “Multilingual” and “Multilingual RE” models respectively. However, it performs on average worse by 6.6% CER compared to the “Multilingual MH”, *i.e.*, when fixing the dataset size. We also show that although using more pre-training data yields better results, training on a variety of languages results in a further improvement, yielding on average a relative CER decrease of 2.6% (comparing the “English-only” to the “Multilingual RE” model). Furthermore, our experiments suggest that the similarity of languages when pre-training and fine-tuning is important, and that pre-training multilingual models yield competitive performance even when fine-tuning on unseen languages, with a relative performance decrease of 5.4% on average (comparing the “Multilingual RE” model to “Multilingual - Unseen”).

CHAPTER 4

BRAVEN: IMPROVING SELF-SUPERVISED PRE-TRAINING FOR SPEECH RECOGNITION

Contents

4.1	Introduction	74
4.2	Method	76
4.2.1	Background: RAVEn	76
4.2.2	BRAVEN	77
4.2.3	Fine-tuning	78
4.3	Experimental Setup	79
4.3.1	Datasets	79
4.3.2	Architecture	79
4.3.3	Data pre-processing	80

4.3.4	Training details	80
4.4	Results	83
4.4.1	Low-resource setting	83
4.4.2	High-resource setting	83
4.4.3	Audio-visual results	84
4.4.4	Ablations	85
4.5	Limitations	85
4.6	Conclusion	86

4.1 Introduction

Visual and auditory speech recognition (VSR / ASR), *i.e.*, the prediction of spoken words from visual and auditory inputs respectively, scale very well with the amount of transcribed data (Makino *et al.*, 2019; Prajwal *et al.*, 2022; Serdyuk *et al.*, 2021, 2022). However, the cost of accurately annotating large-scale datasets can be prohibitive; indeed, the largest publicly available labelled audio-visual dataset has under 500 hours of footage (Afouras *et al.*, 2018b). This issue has spurred significant research into bypassing annotated data requirements.

One line of work along this direction is to use off-the-shelf ASR models to transcribe visual data (lip movement videos) and then train VSR models with a mix of labelled and pseudo-labelled data (Ma *et al.*, 2022, 2023). Similarly, (Afouras *et al.*, 2020) uses the outputs of an ASR model for cross-modal distillation. A different approach is to synthesise lip movements using transcribed audio, which in general is more abundant than transcribed visual data (Liu *et al.*, 2023). Although such approaches obtain impressive results, they still assume access to large-scale transcribed speech data, either to train the ASR model used for pseudo-labelling / distillation, or to assign labels to the corresponding synthesised visual data. It has also been observed that supervised training of VSR models may suffer from optimisation difficulties and may thus require intricate training strategies such as curriculum learning (Haliassos *et al.*, 2023; Ma *et al.*, 2022).

Another line of research, free from these drawbacks, is audio-visual self-supervised learning. In this framework, representations are first learned by exploiting the fine-grained correspondence between the visual and auditory modalities, and then the resulting encoders are fine-tuned on potentially less labelled data for speech recognition (Shi *et al.*, 2022; Haliassos *et al.*, 2023; Baevski *et al.*, 2022; Ma *et al.*, 2021a; Zhu *et al.*, 2023). For example, LiRA (Ma *et al.*, 2021a) learns a visual encoder by predicting audio PASE+ features (Ravanelli *et al.*, 2020). In contrast, AV-HuBERT (Shi *et al.*, 2022) learns audio-visual features by predicting clustering indices generated in multiple stages, initially using MFCC features. VATLM (Zhu *et al.*, 2023) extends AV-HuBERT by incorporating text data into the pre-training process. Recently, student-teacher frameworks have been proposed (Haliassos *et al.*, 2023; Lian *et al.*, 2023) which, given masked inputs, predict contextualised targets generated by momentum teachers that are presented with unmasked inputs. In particular, RAVEn (Haliassos *et al.*, 2023) uses separate video and audio students which regress the outputs of the teacher networks through lightweight Transformer predictors, and learns strong visual and auditory speech representations entirely from raw data.

In this chapter, we extend RAVEn (**R**aw **A**udio-**V**isual **E**ncoders) through modifications that yield better representations by acknowledging the semantic asymmetries between audio and video. We dub our approach Better RAVEn, or BRAVEN. Our enhancements are as follows: (1) We use the average of the outputs of each Transformer encoder block as our targets (Lian *et al.*, 2023), rather than the output of the last block, in order to create smoother targets; (2) We use a shallower predictor for the video student, which encourages the video encoder to better capture the information embedded in the predicted audio targets; (3) We use stronger masking for the audio inputs to address the difference in relative difficulty between VSR and ASR; (4) Finally, we use different loss weights for the audio predictors, which empirically benefits ASR performance.

We find that BRAVEN scales well both with the model size as well as the amount of unlabelled data, consistently achieving state-of-the-art performance across self-supervised methods in comparable settings. We also explore increasing the quantity of unlabelled data beyond what has been used in competing methods by combining LRS3 (Afouras *et al.*, 2018b), Vox-

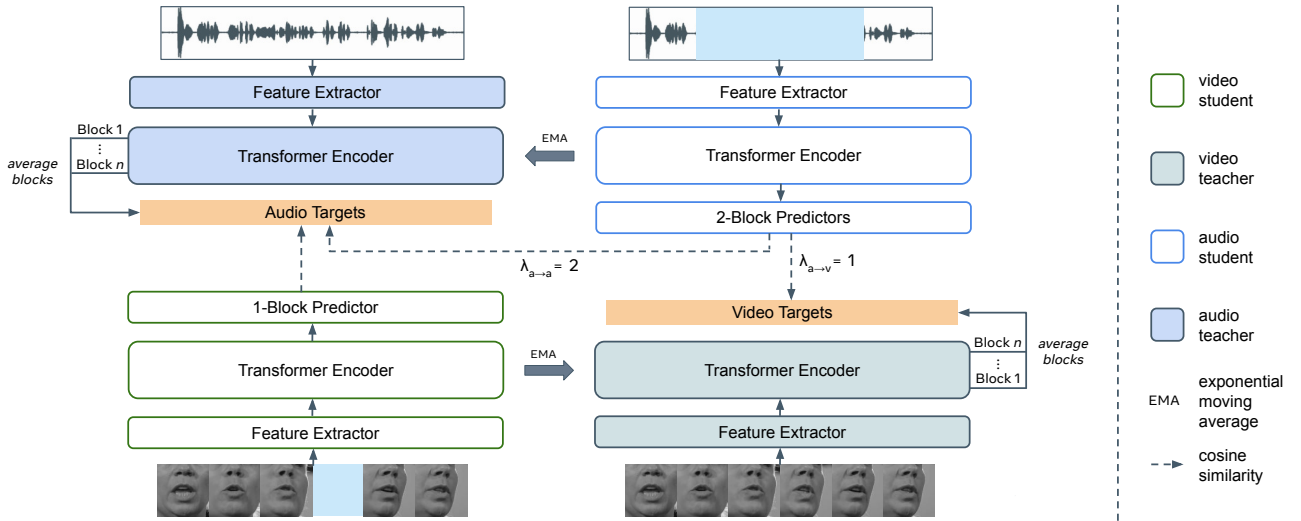


Figure 4.1: **BRAVEN overview**. BRAVEN uses targets that are the average of the outputs of the Transformer encoder blocks of the teacher networks. The video student predicts the audio targets via a 1-block predictor. In contrast, the audio student uses 2-block predictors and uses both a cross- as well as a within-modal loss, whose weight is twice as large as that of the cross-modal loss. The input masking for audio is stronger than the masking applied for video.

celeb2 (Chung *et al.*, 2018), and AVSpeech (Ephrat *et al.*, 2018). We observe that, keeping the amount of labelled data fixed, this leads to further significant improvements, especially for VSR. Notably, BRAVEN-Large trained with around 3,000 hours of unlabelled data and only 30 hours of annotated data achieves 20.0% / 1.7% word error rate (WER) for VSR / ASR on the LRS3 test set, making it competitive with methods trained on orders of magnitude more transcribed data (Makino *et al.*, 2019; Serdyuk *et al.*, 2021).

4.2 Method

4.2.1 Background: RAVEn

RAVEN (Haliassos *et al.*, 2023) learns visual and auditory speech representations by leveraging the correspondence between lip movements and the associated speech waveform, as well as employing masked prediction to encourage the networks to learn context-aware representations. We encourage the reader to re-visit section 3.3.1 if they require a reminder on the high-level overview of this method. We re-define the notation of the losses below to be consistent with

the rest of this chapter. The losses for the video student, $\mathcal{L}_v$, and audio student, $\mathcal{L}_a$, are given as

$$\mathcal{L}_v = \mathcal{L}^{v \rightarrow a}, \quad \mathcal{L}_a = \lambda_{a \rightarrow v} \mathcal{L}^{a \rightarrow v} + \lambda_{a \rightarrow a} \mathcal{L}^{a \rightarrow a}, \quad (4.1)$$

where $\mathcal{L}^{v \rightarrow a}$, $\mathcal{L}^{a \rightarrow v}$, and $\mathcal{L}^{a \rightarrow a}$ denote the video-to-audio, audio-to-video, and audio-to-audio prediction losses, respectively, and $\lambda_{a \rightarrow v}$ and $\lambda_{a \rightarrow a}$ are scaling factors, which are both set to 1.

4.2.2 BRAVEN

In the following, we describe in detail each of BRAVEN’s design improvements over RAVEN (see Figure 4.1).

First, BRAVEN uses the mean of the outputs of all Transformer blocks rather than only the final block’s output (Lian *et al.*, 2023), which is possible due to the feature size homogeneity across the Transformer architecture. This averaging operation likely results in smoother and higher-quality targets, which can aid the training dynamics (Grill *et al.*, 2020). Moreover, we apply instance normalisation (Ulyanov *et al.*, 2016) to the averaged targets to prevent representation collapse (Caron *et al.*, 2021). In RAVEN, the layer normalisation (Ba *et al.*, 2016) that follows the final block in the Transformer architecture serves a similar purpose.

Second, BRAVEN uses asymmetric predictor depths. RAVEN uses two-block Transformer predictors (with attention dimension 512), shown to work optimally *when using the same design for both video and audio students*. BRAVEN instead uses a *one-block* Transformer predictor for the video student (which predicts the audio targets), while retaining the original design of the audio predictors. The intuition is that audio is more relevant to speech recognition than video, and thus a shallower predictor leads to visual representations that more closely match the information in the audio targets.

Third, BRAVEN applies stronger masking to the audio inputs. Each corresponding video frame index has a 40 % probability of being picked as the start of a mask for audio, while a probability

of 20 % is used for video¹. In contrast, RAVEEn uses a probability of 20 % for both modalities, which was chosen based on the assumption that *both modalities would equally benefit from the same masking strategy*. BRAVEEn’s masking asymmetry is beneficial likely due to the difference in difficulty between VSR and ASR: Stronger masking for audio leads to more context-aware representations, but for video it may make the pretext task overly difficult.

Finally, BRAVEEn uses asymmetric loss weights. In RAVEEn, the weights of the two losses associated with the audio student are equal. BRAVEEn instead uses a larger weight for the audio-to-audio ($\lambda_{a \rightarrow a} = 2$) than the audio-to-video loss ($\lambda_{a \rightarrow v} = 1$), which we empirically find leads to improvements for ASR.

4.2.3 Fine-tuning

After pre-training, all networks are discarded except for the video and audio student encoders, which are fine-tuned for VSR and ASR, following (Haliassos *et al.*, 2023). We use the CTC / attention loss with a CTC weight of 0.1, which is the same weight used during decoding. We use a beam search size of 40, and our vocabulary is constructed using 1,000 SentencePiece (Kudo and Richardson, 2018) subword units. We also experiment with self-training: The fine-tuned ASR model is used to pseudo-label the unlabelled data, which is then used for fine-tuning the BRAVEEn pre-trained encoders (Shi *et al.*, 2022; Haliassos *et al.*, 2023; Zhu *et al.*, 2023). We also present results when using the language model from (Ma *et al.*, 2022, 2023), incorporated through shallow fusion.

¹Then, the three consecutive video and 1,920 corresponding audio samples are zeroed out (since a video frame corresponds to 640 audio samples).

	Base	Base+	Large
Number of parameters (M)	41	93	328
Number of blocks	12	12	24
Attention dimension	512	768	1024
Number of attention heads	8	12	16
MLP size	2048	3072	4096

Table 4.1: Configuration of the transformer encoders.

4.3 Experimental Setup

4.3.1 Datasets

We use the following datasets for pre-training: LRS3 (Afouras *et al.*, 2018b), VoxCeleb2 (Chung *et al.*, 2018), and AVSpeech (Ephrat *et al.*, 2018). LRS3 contains 433 hours of audio-visual speech. We use English-only versions of VoxCeleb2 and AVSpeech (Shi *et al.*, 2022; Ma *et al.*, 2022), which comprise 1,326 and 1,323 hours of footage, respectively. We refer to a combination of LRS3 with VoxCeleb2 as LRS3+Vox2. When additionally using AVSpeech, we refer to the resulting dataset as LRS3+Vox2+AVS. For fine-tuning, we use either the 30-hour “trainval” set of LRS3 (“low-resource” setting) or the full 433-hour LRS3 dataset (“high-resource” setting). Details on the datasets can be found in Section 2.1.1.

4.3.2 Architecture

Each encoder consists of a modality-specific, convolutional feature extractor followed by a Transformer encoder (Vaswani *et al.*, 2017; Haliassos *et al.*, 2023). The feature extractor for video is a 2D ResNet-18 (He *et al.*, 2016) with a 3D stem (Stafylakis and Tzimiropoulos, 2017), while for audio it is a 1D ResNet-18, as in (Haliassos *et al.*, 2023). We use three different Transformer encoder sizes: Base, Base+, and Large. Details of their configuration are given in Table 4.1. We note that Base+ and Large are model sizes used in related works, such as (Shi *et al.*, 2022) and (Lian *et al.*, 2023). Although these works share the weights of the Transformer encoder for video and audio during pre-training, they separately fine-tune the models for VSR

and ASR, resulting in separate sets of parameters.

For high-resource fine-tuning, we use Transformer decoders with 6 and 9 blocks for Base(+) and Large, respectively. The decoder dimensions (*i.e.*, width) match the corresponding encoders. For low-resource, we use a decoder with 6 blocks and with hidden size, MLP size, and attention heads equal to 256, 2048, and 4, respectively, to avoid overfitting.

4.3.3 Data pre-processing

We follow the common protocol from (Shi *et al.*, 2022; Haliassos *et al.*, 2023; Ma *et al.*, 2022) for pre-processing. We follow the same process as sections 2.2.2 and 2.2.1 for video and audio.

4.3.4 Training details

We use the AdamW (Loshchilov and Hutter, 2019) optimiser with a cosine decay learning scheduler. We pre-train the networks for 150 epochs with a warmup of 40 epochs for LRS3 and of 30 epochs when pre-training with more data. The learning rate is 3×10^{-3} when training BRAVE_n-Base and Base+, and it is 2×10^{-3} and 1×10^{-3} when training BRAVE_n-Large on LRS3+Vox2 and LRS3+Vox2+AVS, respectively. The weight decay is set to 0.04. We use drop path (Huang *et al.*, 2016) with a default value of 0.05, except when training the Large model on LRS3+Vox2, where it is set to 0.1. We use a maximum of 2,400 video frames per GPU for the Base and Base+ models and 900 frames per GPU for BRAVE_n-Large. For video augmentations, we horizontally flip the video with 50 % probability and randomly crop them to size 88×88 . We do not augment the audio. For fine-tuning, we use the protocol from (Haliassos *et al.*, 2023).

Method	Encoder	LM	Unlab hrs	Lab hrs	WER (%)	
					VSR	ASR
<i>supervised (includes non-publicly available data)</i>						
V2P (Shillingford <i>et al.</i> , 2019)	RNN	✓	-	3,886	55.1	-
RNN-T (Makino <i>et al.</i> , 2019)	RNN	✗	-	31,000	33.6	4.8
VTP (Prajwal <i>et al.</i> , 2022)	Transf	✓	-	2,676	30.7	-
ViT3D-TM (Serdyuk <i>et al.</i> , 2021)	Transf	✗	-	90,000	25.9	2.3
ViT3D-CM (Serdyuk <i>et al.</i> , 2022)	Conf	✗	-	90,000	17.0	1.6
Base(+) models						
AV-HuBERT (Shi <i>et al.</i> , 2022)	Transf	✗	403	30	51.8	4.9
VATLM [†] (Zhu <i>et al.</i> , 2023)	Transf	✗	403	30	48.0	-
RAVEN [‡] (Haliassos <i>et al.</i> , 2023)	Transf	✗	403	30	47.0	4.7
AV-data2vec (Lian <i>et al.</i> , 2023)	Transf	✗	403	30	45.2	4.4
BRAVEN [‡]	Transf	✗	403	30	43.4	4.0
Base(+) models						
AV-HuBERT (Shi <i>et al.</i> , 2022)	Transf	✗	1,729	30	46.1	4.6
VATLM [†] (Zhu <i>et al.</i> , 2023)	Transf	✗	1,729	30	42.6	-
RAVEN [‡] (Haliassos <i>et al.</i> , 2023)	Transf	✗	1,729	30	40.2	3.8
AV-data2vec (Lian <i>et al.</i> , 2023)	Transf	✗	1,729	30	37.8	3.7
BRAVEN	Transf	✗	1,729	30	35.1	3.0
Large models						
AV-HuBERT (Shi <i>et al.</i> , 2022)	Transf	✗	1,729	30	32.5	2.9
VATLM [†] (Zhu <i>et al.</i> , 2023)	Transf	✗	1,729	30	31.6	-
RAVEN (Haliassos <i>et al.</i> , 2023)	Transf	✗	1,729	30	32.5	2.7
AV-data2vec (Lian <i>et al.</i> , 2023)	Transf	✗	1,729	30	30.8	2.7
AV-HuBERT w/ ST (Shi <i>et al.</i> , 2022)	Transf	✗	1,729	30	28.6	-
VATLM [†] w/ ST (Zhu <i>et al.</i> , 2023)	Transf	✗	1,729	30	27.6	-
RAVEN w/ ST (Haliassos <i>et al.</i> , 2023)	Transf	✗	1,729	30	24.8	2.3
BRAVEN	Transf	✗	1,729	30	30.8	2.3
BRAVEN	Transf	✗	3,052	30	24.8	2.1
BRAVEN w/ ST	Transf	✗	3,052	30	21.3	1.9
BRAVEN w/ ST	Transf	✓	3,052	30	20.0	1.7

Table 4.2: **LRS3 results: low-resource.** “Unlab hrs”/“lab hrs” denote unlabelled hours/labelled hours. Self-supervised methods use labelled data for pre-training (without labels) along with unlabelled data (if any). “LM” denotes language model. “ST” denotes self-training. [†]Uses an additional 3,846/452 hours of audio/audio-text data. [‡]Uses the (smaller) Base model.

Method	Encoder	LM	Unlab hrs	Lab hrs	WER (%)	
					VSR	ASR
<i>semi-supervised (uses external models for pseudo-labelling)</i>						
KD+CTC (Afouras <i>et al.</i> , 2020)	CNN	✓	344	433	59.8	-
CM-aux (Ma <i>et al.</i> , 2022)	Conf	✓	641	818	31.5	-
Auto-AVSR (Ma <i>et al.</i> , 2023)	Conf	✓	2,630	818	19.1	1.0
SynthVSR Large (Liu <i>et al.</i> , 2023)	Conf	✓	2,630	4,090*	16.9	-
Base(+) models						
AV-HuBERT (Shi <i>et al.</i> , 2022)	Transf	✗	-	433	44.0	-
RAVEN [‡] (Haliassos <i>et al.</i> , 2023)	Transf	✗	-	433	39.1	2.2
AV-data2vec (Lian <i>et al.</i> , 2023)	Transf	✗	-	433	39.0	2.0
BRAVEN [‡]	Transf	✗	-	433	36.0	1.9
Base(+) models						
AV-HuBERT (Shi <i>et al.</i> , 2022)	Transf	✗	1,326	433	34.8	2.0
VATLM [†] (Zhu <i>et al.</i> , 2023)	Transf	✗	1,326	433	34.2	-
RAVEN [‡] (Haliassos <i>et al.</i> , 2023)	Transf	✗	1,326	433	33.1	1.9
AV-data2vec (Lian <i>et al.</i> , 2023)	Transf	✗	1,326	433	32.9	1.7
BRAVEN	Transf	✗	1,326	433	28.8	1.4
Large models						
AV-HuBERT (Shi <i>et al.</i> , 2022)	Transf	✗	1,326	433	28.6	1.3
VATLM [†] (Zhu <i>et al.</i> , 2023)	Transf	✗	1,326	433	28.4	-
RAVEN (Haliassos <i>et al.</i> , 2023)	Transf	✗	1,326	433	27.8	1.4
AV-data2vec (Lian <i>et al.</i> , 2023)	Transf	✗	1,326	433	28.5	1.4
u-HuBERT (Hsu and Shi, 2022)	Transf	✗	1,326	433	27.2	1.4
AV-HuBERT w/ ST (Shi <i>et al.</i> , 2022)	Transf	✗	1,326	433	26.9	-
VATLM [†] w/ ST (Zhu <i>et al.</i> , 2023)	Transf	✗	1,326	433	26.2	-
RAVEN w/ ST (Haliassos <i>et al.</i> , 2023)	Transf	✗	1,326	433	24.4	1.4
BRAVEN	Transf	✗	1,326	433	26.6	1.2
BRAVEN	Transf	✗	2,649	433	23.6	1.2
BRAVEN w/ ST	Transf	✗	2,649	433	20.9	1.2
BRAVEN w/ ST	Transf	✓	2,649	433	20.1	1.1

Table 4.3: **LRS3 results: high-resource.** “Unlab hrs”/“lab hrs” denote unlabelled hours/labelled hours. Self-supervised methods use labelled data for pre-training (without labels) along with unlabelled data (if any). “LM” denotes language model. “ST” denotes self-training. *Includes synthetic visual data generated using labelled audio. †Uses an additional 3,846/452 hours of audio/audio-text data. ‡Uses the (smaller) Base model.

4.4 Results

4.4.1 Low-resource setting

We provide results for the low-resource setting in Table 4.2. Using BRAVE_n-Base and LRS3 as the pre-training dataset, we achieve better VSR and ASR word error rate (WER) than the previous state of the art obtained by AV-data2vec (43.4 % / 4.0 % vs 45.2 % / 4.4 % for VSR / ASR), despite using approximately half the number of parameters during inference. Using LRS3+Vox2 for pre-training, BRAVE_n-Base+ obtains 35.1 % / 3.0 % WER for VSR / ASR, which represents the state of the art for this setting too. Increasing the model size to Large improves the results to 30.8 % and 2.3 % WER for VSR and ASR respectively, demonstrating BRAVE_n’s strong scalability. We are also the first to experiment with pre-training on LRS3+Vox2+AVS, consisting of 3,082 hours of unlabelled data. This increase in pre-training data results in significant WER improvement for VSR (30.8 %→24.8 %) and modest improvement for ASR (2.3 %→2.1 %). We believe we have not reached saturation yet, and increasing pre-training data further will yield even more improvements. Finally, using self-training and a language model during inference leads to 20.0 % / 1.7 % WER for VSR / ASR. We point out that our results with only *30 hours* of labelled data (and no external pre-trained ASR model) are competitive with supervised methods that use orders of magnitude of more labelled data (Makino *et al.*, 2019; Serdyuk *et al.*, 2021).

4.4.2 High-resource setting

The results for the high-resource setting are shown in Table 4.3. In each setting, BRAVE_n achieves state-of-the-art performance among the self-supervised methods. Our best results for the high-resource setting are 20.1 % / 1.1 % for VSR / ASR. Interestingly, BRAVE_n-Large coupled with self-training and a language model yields similar VSR results ($\sim 20\%$ WER) in both low- and high-resource settings, but better results are obtained in the high-resource setting for ASR. This suggests that high-quality transcriptions are more important for ASR

	Clean	SNR (dB)		
		5	0	-5
AV-data2vec (AV)	4.2	-	-	-
RAVE _n (AV)	4.7	14.3	18.2	58.0
BRAVE _n (A)	4.0	15.6	24.6	99.0
BRAVE _n (AV)	4.0	12.4 ↓3.2	15.0 ↓9.6	48.5 ↓50.5

Table 4.4: **Audio-visual results.** We show low-resource results (WER %) (Base models) with different signal-to-noise ratio (SNR).

than VSR. Furthermore, our results are competitive with semi-supervised methods that use external ASR models trained on tens of thousands of hours of audio data (Ma *et al.*, 2023; Liu *et al.*, 2023). All in all, our results suggest that BRAVE_n can effectively translate unlabelled audio-visual data into strong speech recognition performance for both modalities.

4.4.3 Audio-visual results

Although not the focus of this work, we now provide results for audio-visual speech recognition using the late-fusion method proposed in (Ma *et al.*, 2021c). To that end, we append a fusion module that concatenates the features from the video and audio encoders and applies a two-layer MLP with hidden dimension 1024. We transfer the weights from the VSR and ASR encoders and then train the fusion model, CTC layer, and decoder for 200 epochs, keeping the encoder weights frozen.

Results for the low-resource setting with the Base model and different levels of audio babble noise from the NOISEX corpus (Varga and Steeneken, 1993) are given in Table 4.4. It is evident that as the audio noise level increases, so does the difference in performance between the audio-only and audio-visual models. BRAVE_n outperforms RAVE_n at all noise levels and achieves better WER than AV-data2vec, *even when BRAVE_n uses only the audio modality*.

	WER (%)	
	VSR	ASR
RAVE _n	47.0	4.7
+ average blocks	45.3 ↓1.7	4.6 ↓0.1
+ shallower video predictor	44.1 ↓1.2	4.6 =
+ stronger audio masking	43.5 ↓0.6	4.2 ↓0.4
+ different loss weights = BRAVE_n	43.4 ↓0.1	4.0 ↓0.2
BRAVE _n + audio mask probability 0.6	44.4 ↑1.0	4.2 ↑0.2
BRAVE _n + averaging last 6 blocks	46.2 ↑2.8	4.1 ↑0.1

Table 4.5: **Ablations.** We show results for the low-resource setting using the Base model.

4.4.4 Ablations

We conduct an ablation study to show the effect of each design choice on the VSR and ASR performance (see Table 4.5). Each new addition improves the VSR and / or ASR performance. We observe that averaging targets and using a shallower video predictor has a larger effect on VSR than ASR, while stronger audio masking and different loss weights have a more significant impact on ASR. Furthermore, using even stronger audio masking or using the average of the last 6 Transformer blocks as targets lead to worse performance.

4.5 Limitations

Our work focuses on improving self-supervised methodologies for speech recognition. While experiments are conducted exclusively on English data, the extent to which the proposed approach generalizes to non-English languages remains unexplored; evaluating cross-lingual generalization, either in zero-shot settings or through pre-training and fine-tuning on multilingual data, is an important direction for future work. Furthermore, the current approach relies on separate encoders for the visual and auditory modalities, which require independent fine-tuning and hyperparameter configurations, resulting in a more complex training pipeline and increased computational requirements, such as GPU usage and storage for multiple checkpoints. Investigating unified encoder architectures that support ASR, VSR, and AVSR within a single fine-tuning stage could reduce this complexity and improve efficiency.

4.6 Conclusion

We have presented BRAVEEn, a self-supervised framework for learning visual and auditory speech representations from audio-visual data. We propose some design improvements to the recent RAVEEn method, which cumulatively have a significant impact on the downstream VSR and ASR performance and achieve state-of-the-art results for audio-visual self-supervised methods in various settings. Furthermore, we experiment with doubling the amount of unlabelled data during pre-training as used in other self-supervised works and observe strong scaling behaviour, even when fine-tuning with only 30 hours of labelled data. Our work provides compelling evidence that readily accessible unlabelled audio-visual data can effectively substitute costly annotated samples.

CHAPTER 5

FLASHLIPS: 100-FPS MASK-FREE LATENT LIP-SYNC USING RECONSTRUCTION

Contents

5.1	Introduction	88
5.2	Related Work	92
5.2.1	Audio-driven Portrait Animation	92
5.2.2	Audio-driven Lip Synchronization	93
5.3	Method	94
5.3.1	Stage 1: Latent Visual Editor	94
5.3.2	Stage 2: Audio-to-Lips with Flow Matching	98
5.3.3	Architecture Details	99
5.4	Experiments	101
5.4.1	Datasets	101
5.4.2	Training Details	102

5.4.3	Evaluation Metrics	104
5.4.4	Quantitative Evaluation	105
5.4.5	Qualitative Evaluation	106
5.4.6	User Study	107
5.4.7	Mask Removal: Quantitative Impact	107
5.5	Ablation Study	109
5.6	Limitations	110
5.7	Conclusion	111

Our thesis is focused on leveraging unlabelled audio-visual data to develop efficient and accurate systems. The previous chapters concentrated on applying this paradigm to speech recognition using self-supervised learning. In this chapter, we pursue the same overarching goal but shift to a different task: lip synchronization. Whereas speech recognition requires converting spoken audio into text, lip-sync generation demands animating a face conditioned on the speaker’s audio, which in turn calls for a different modeling approach.

5.1 Introduction

Lip synchronization (lip-sync) is the task of regenerating realistic mouth movements that match audio while preserving identity, expression, head pose, background, and overall fidelity of a talking-person video. It has a transformative impact across domains — from automating film/TV dubbing and breaking language barriers, to creating expressive animations and lifelike digital avatars (Zhang *et al.*, 2023c; Zhen *et al.*, 2023). The central challenge is synthesizing photorealistic, temporally stable lip motions precisely synchronized with speech.

Audio-driven facial generation is closely related: it animates the full face from a reference image and a driving audio (Zhou *et al.*, 2019; Vougioukas *et al.*, 2019; Xu *et al.*, 2024a; Chen *et al.*, 2025b; Wang *et al.*, 2024). Compared to full-face generation, lip-sync is more controllable and



Figure 5.1: **FlashLips results.** Selected results of source and driver pairs, generated using our transformer-based model.

efficient: it edits only the mouth while reusing identity, pose, and background from the target video, crucial for dubbing/localization.

The pursuit of high fidelity has produced a spectrum of deep learning approaches (Goodfellow *et al.*, 2016). Early successes were driven by GANs (Goodfellow *et al.*, 2014; Prajwal *et al.*, 2020b; Guan *et al.*, 2023), which can yield sharp frames but are notoriously difficult to train and sensitive to hyperparameters (Salimans *et al.*, 2016; Srivastava *et al.*, 2017). More recently, iterative generative models – particularly diffusion – have set a strong bar for visual quality in both general face animation (Du *et al.*, 2025; Shen *et al.*, 2023; Stypułkowski *et al.*, 2024; Zhang *et al.*, 2024) and task-specific lip-sync (Bigioi *et al.*, 2024; Liu *et al.*, 2024; Mukhopadhyay *et al.*, 2024). However, diffusion requires sequential inference (multiple denoising steps), compounding cost and often prompting additional pre/post-processing such as explicit mouth masks or alignment to canonical templates (Bigata *et al.*, 2025b; Li *et al.*, 2025), which complicates real-time deployment and adds engineering overhead.

In this work, we take a step back and question the necessity of iterative *visual* generators for a highly conditioned task like lip-sync. We argue that with sufficient context – a reference identity, a target frame, and precise lips-pose cues – a powerful *deterministic* image update can be learned without adversarial objectives or diffusion schedules.

We introduce a lightweight, two-stage framework that separates *control* from *rendering* in the spirit of two-stage designs of (Liu *et al.*, 2024).

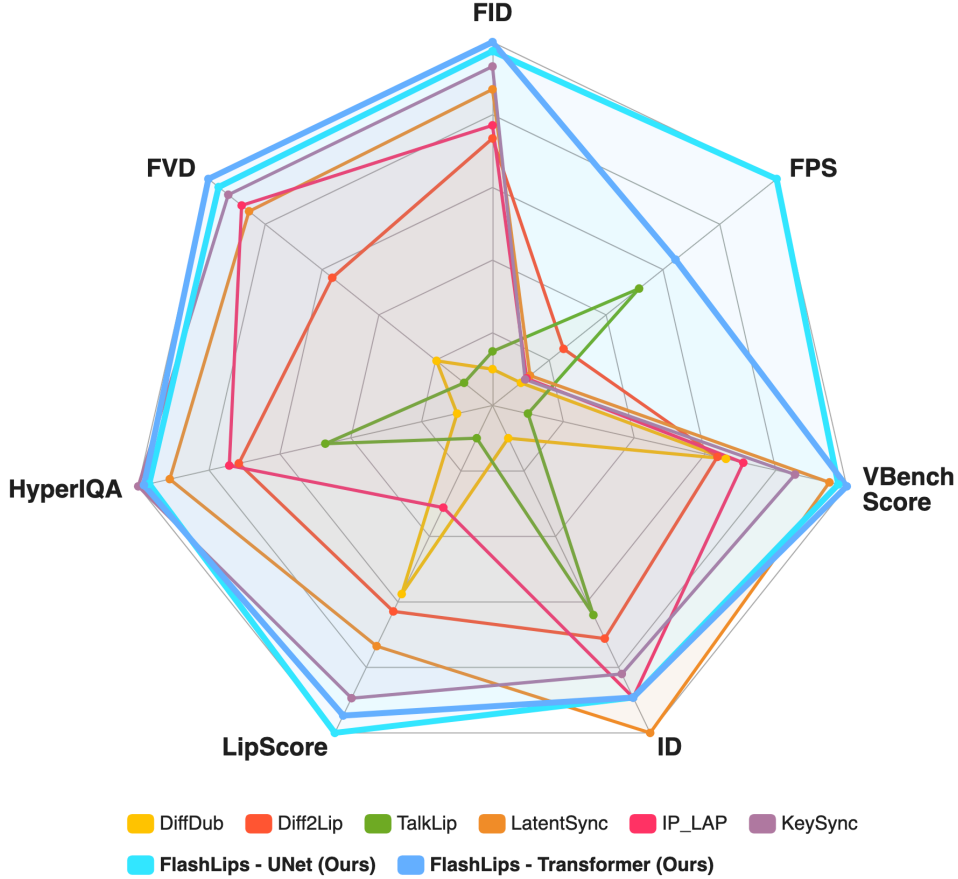


Figure 5.2: **Visualization of quantitative evaluation.** Comparison of eight different lip-sync models in the cross-audio setting on seven key metrics. All results are normalized, with the best-performing model scaled to the outer edge, and the worst scaled towards the center.

Stage 1: Latent Visual Editor. We start with a compact *one-step* editor operating in VAE latent space (Podell *et al.*, 2023). Given a reference image, a target frame, and a low-dimensional *lips-pose vector*, it reconstructs the edited frame in a single feed-forward pass using *reconstruction losses only* - no adversarial training or diffusion.

Our final editor runs *without* explicit mouth masks at inference. After reconstruction training,

we self-refine by synthesizing mouth-altered variants and fine-tuning on symmetric *source* $\leftrightarrow$ *changed* pairs, which focuses edits on the lips while preserving the rest of the frame without any external segmentation. Prior work established mask-free feasibility (Peng *et al.*, 2025); we instead create the supervision *on-the-fly* with the editor itself, avoiding dataset-constructed pseudo-pairs.

Stage 2: Audio-to-Lips. *Stage 2* connects audio to the visual editor via an audio-to-lips transformer that predicts lips-pose vectors from speech. A key design principle is to *disentangle* the control space so that it carries *pose* information only, i.e., what the lips should do, while appearance (teeth, lip color, skin tone, jawline) and scene details are sourced from the reference and target frames in Stage 1. This mirrors what can be reliably inferred from audio and keeps Stage 2 lightweight and stable to train. Overloading audio with appearance factors makes learning harder and harms generalization; our disentangled control avoids this by construction. Conditioning on wav2vec 2.0 features (Baevski *et al.*, 2020b) and training with *flow-matching* (Lipman *et al.*, 2022; Liu *et al.*, 2022) yields smooth control latents that drive the editor.

Our contributions:

- **Real-time performance:** >100 FPS on a single NVIDIA H100 80GB HBM3 that matches or exceeds larger, slower baselines in terms of lip-sync accuracy and quality (Fig. 5.2).
- **Deterministic, one-step feasibility:** For a highly conditioned task (such as lip-sync), reconstruction-only training might be sufficient, removing the need for adversarial or iterative generators – *no GANs or diffusion*.
- **Mask-free self-refinement:** no explicit mouth masks at inference; fewer mouth artifacts and a simpler pipeline.
- **Disentangled audio-to-pose:** flow-matching transformer on wav2vec 2.0 separates *what* to render from *how* to render, supporting modular, per-component control.

5.2 Related Work

5.2.1 Audio-driven Portrait Animation

Audio-driven portrait animation synthesizes talking-head videos from speech (Tian *et al.*, 2025; Ji *et al.*, 2025; Jiang *et al.*, 2025), yet its objectives differ fundamentally from lip synchronization. It follows an image-to-video paradigm that freely modifies head pose and facial expressions without constraining the output to match a specific input video. In contrast, lip synchronization operates in a video-to-video setting, modifying only the mouth region while preserving all other facial details, effectively editing the original video.

Early GAN-based approaches employed temporal and task-specific discriminators to animate faces (Zhou *et al.*, 2019; Vougioukas *et al.*, 2019, 2018), later incorporating head-pose modeling to improve the results, though often introducing visual artifacts (Zhou *et al.*, 2021; Zhang *et al.*, 2023b). Diffusion-based methods further enhanced temporal coherence and perceptual quality (Xu *et al.*, 2024a; Stypułkowski *et al.*, 2024; Jiang *et al.*, 2025), sometimes leveraging facial landmarks or 3D meshes (Wei *et al.*, 2024; Zhang *et al.*, 2023a), although the latter can lead to unrealistic animations. Frameworks with a two stage inference (Bigata *et al.*, 2025a) first generate intermediate keyframes and then interpolate between them to achieve smoother, temporally consistent motion. Despite producing plausible facial motion, many of these models suffer from high inference latency. To address this, recent works (Xu *et al.*, 2024b; Ki *et al.*, 2025) train small audio-to-latent diffusion models that drive pre-trained latent-to-video decoders, achieving near real-time performance.

Although these methods have advanced portrait animation, their freedom to alter head pose and expression makes them unsuitable for lip synchronization, motivating dedicated methods that preserve identity and facial consistency.

5.2.2 Audio-driven Lip Synchronization

Lip synchronization aims to modify only the mouth region to match the input audio while preserving head pose, identity, and other facial expressions. Wav2Lip (Prajwal *et al.*, 2020b) popularized SyncNet-based (Chung and Zisserman, 2016) supervision to ensure reliable audio-visual alignment. Subsequent works such as DINet (Zhang *et al.*, 2023c), IP-LAP (Zhong *et al.*, 2023), and ReSyncer (Guan *et al.*, 2025) enhance realism through spatial deformation of reference features, intermediate landmark prediction, or the use of 3D priors. Other works such as StyleSync (Guan *et al.*, 2023) and StyleLipSync (Ki and Min, 2023) adopt StyleGAN-inspired architectures (Karras *et al.*, 2020), while TalkLip (Wang *et al.*, 2023) leverages a lip-reading expert within a contrastive learning framework to enhance lip-speech synchronization.

Diffusion-based approaches further enhance temporal consistency and perceptual quality (Bigioi *et al.*, 2024; Liu *et al.*, 2024; Mukhopadhyay *et al.*, 2024; Peng *et al.*, 2025). Latent diffusion frameworks such as LatentSync (Li *et al.*, 2025) and SayAnything (Ma *et al.*, 2025) synthesize lip-synced frames directly from audio without intermediate motion conditioning. MuseTalk (Zhang *et al.*, 2025) refines synchronization by selecting reference frames with similar head poses, and KeySync (Bigata *et al.*, 2025b) mitigates temporal inconsistency and leakage through a keyframe-interpolation approach and careful masking strategy.

Nevertheless, existing approaches still face notable limitations. GAN-based methods often suffer from visual artifacts and unstable training, while diffusion-based models, although capable of high-quality synthesis, remain computationally expensive and generally too slow for real-time inference. Moreover, many pipelines depend on extensive pre-processing (e.g., face alignment or intermediate motion estimation), which can introduce artifacts and reduce flexibility. These challenges motivate our work: a simple and efficient framework for high-quality lip synchronization that eliminates iterative generation and heavy pre-processing, enabling faster-than-real-time inference with high-resolution outputs.

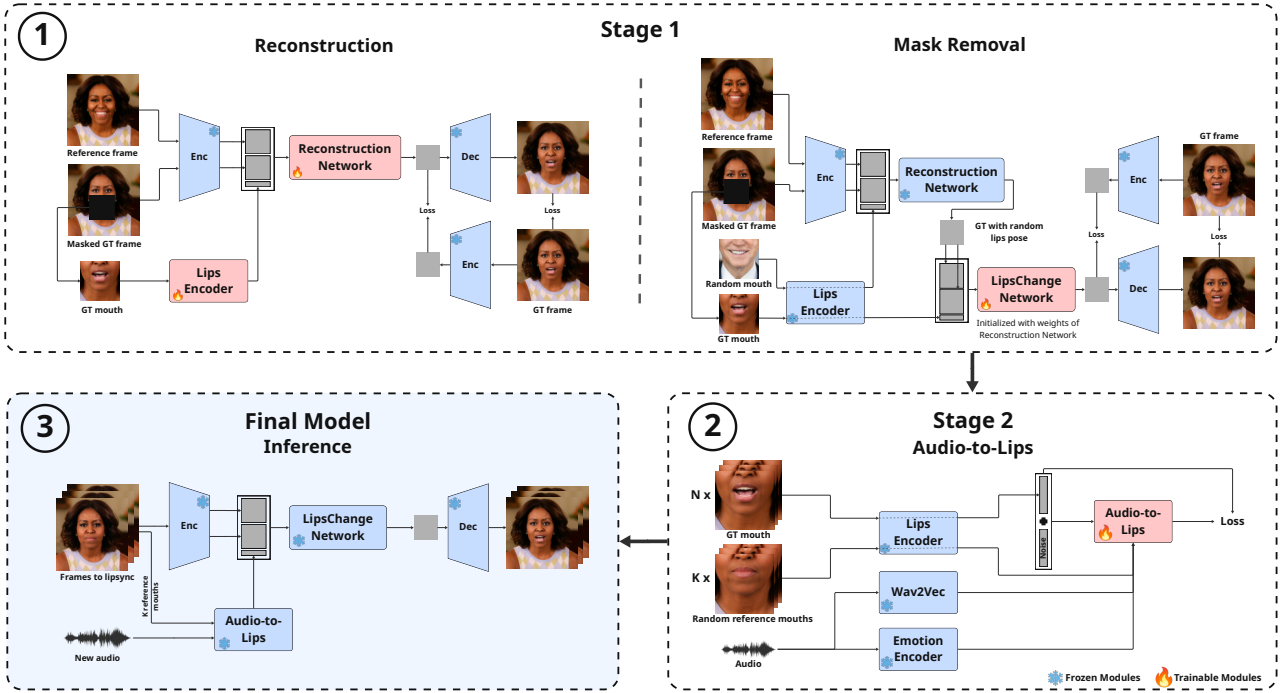


Figure 5.3: **Overview of FlashLips.** **Stage 1** trains a one-step latent-space editor: first via masked reconstruction, then via a mask-free self-refinement step that learns to localize edits without segmentation. **Stage 2** trains an audio-to-lips model that predicts the lips-pose vector used in Stage 1. At inference, predicted lip poses drive the LipsChange network to produce lip-synced frames in a single pass.

5.3 Method

We propose a two-stage framework for lip synchronization (Fig. 5.3). **Stage 1** is a fast, deterministic editor that produces high-quality lip-synced frames in a single forward pass. **Stage 2** is an audio-to-pose transformer that predicts low-dimensional lip poses from speech and drives Stage 1. Stage 1 is trained *only* with reconstruction objectives (no adversarial training or diffusion); Stage 2 uses a flow-matching objective.

5.3.1 Stage 1: Latent Visual Editor

Reconstruction and Lips Encoder

Stage 1 is trained per frame in latent space following (Rombach *et al.*, 2022). Let $\mathbf{x}_{\text{src}}, \mathbf{x}_{\text{ref}} \in \mathbb{R}^{C \times H \times W}$ be the frame to edit (“source”) and a reference frame from the same video, sampled

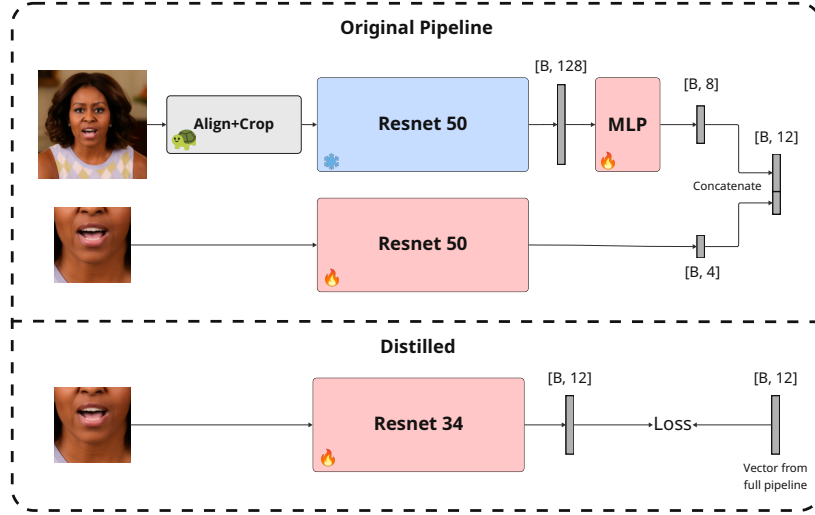


Figure 5.4: **Lips encoder.** A frozen expression encoder with an MLP projector and a mouth-crop CNN produce an 8D+4D lips vector. A distilled ResNet-34 replicates this mapping on inference.

t frames apart, respectively. During reconstruction training we apply a mouth-region mask to $\mathbf{x}_{\text{src}}$, yielding $\mathbf{x}_{\text{masked}}$.

We encode $\mathbf{x}_{\text{src}}$, $\mathbf{x}_{\text{masked}}$, and $\mathbf{x}_{\text{ref}}$ with the SDXL VAE (Kingma and Welling, 2014; Podell *et al.*, 2023) to obtain latents $\mathbf{z}_{\text{src}}, \mathbf{z}_{\text{masked}}, \mathbf{z}_{\text{ref}} \in \mathbb{R}^{C_\ell \times H_\ell \times W_\ell}$. A small trainable *reference* backbone f_{ref} projects the reference latent, $\bar{\mathbf{z}}_{\text{ref}} = f_{\text{ref}}(\mathbf{z}_{\text{ref}})$, so the model can better adapt identity features. We also use a lips-pose representation $\mathbf{z}_{\text{lips}} \in \mathbb{R}^M$ that encodes lips configuration. It is expanded spatially by replication, $\mathbf{z}_{\text{lips expanded}} \in \mathbb{R}^{M \times H_\ell \times W_\ell}$.

The network’s input is the channel-wise concatenation:

$$\mathbf{z}_{\text{input}} = \text{Concat} [\mathbf{z}_{\text{masked}}, \bar{\mathbf{z}}_{\text{ref}}, \mathbf{z}_{\text{lips expanded}}]. \quad (5.1)$$

It predicts a latent residual towards the ground-truth edit. With:

$$\mathbf{z}_{\text{target}} = \mathbf{z}_{\text{src}} - \mathbf{z}_{\text{masked}}, \quad (5.2)$$

the model outputs $\hat{\mathbf{z}}_{\text{target}}$ and forms:

$$\hat{\mathbf{z}}_{\text{src}} = \mathbf{z}_{\text{masked}} + \hat{\mathbf{z}}_{\text{target}}. \quad (5.3)$$

Decoding with the frozen VAE yields $\hat{\mathbf{x}}_{\text{src}} = \text{VAE}_{\text{decoder}}(\hat{\mathbf{z}}_{\text{src}})$.

Lips-Pose Representation. We design the representation to carry *only* lip/jaw configuration to predict it more easily from audio. A frozen expression encoder (Drobyshev *et al.*, 2024) with a small MLP yields $\mathbf{z}_{\text{lips}}^{\text{main}} \in \mathbb{R}^M$. In parallel, a lightweight CNN on a mouth crop predicts a small residual $\mathbf{z}_{\text{lips}}^{\text{add}}$. The final control is $\mathbf{z}_{\text{lips}} = \mathbf{z}_{\text{lips}}^{\text{main}} + \mathbf{z}_{\text{lips}}^{\text{add}}$. For faster inference, we distill these two models into a compact image encoder (e.g., a ResNet-34 head (He *et al.*, 2016)) that predicts $\mathbf{z}_{\text{lips}}$ directly from an RGB face crop; see Fig. 5.4. Stage 2 is trained to predict the same vector from audio.

Mask Removal via Self-Refinement

After the reconstruction editor R_ϕ converges, we synthesize lip-altered counterparts for real frames. Given a real frame S and a sampled lips-pose vector $\mathbf{z}_{\text{lips}}$, we produce $\tilde{S} = R_\phi(S; \mathbf{z}_{\text{lips}})$ and form symmetric pseudo-pairs $(S \rightarrow \tilde{S})$ and $(\tilde{S} \rightarrow S)$. We initialize the *LipsChange* editor $L_\theta \leftarrow R_\phi$ and fine-tune it on these pairs using the same objective as in Section 5.3.1. This teaches the model to localize edits to the lips and preserve surrounding regions, eliminating the need for external segmentation (Fig. 5.3). At test time, L_θ runs *without* any explicit mouth masks.

Why two directions?

- $(S \rightarrow \tilde{S})$ (**real** $\rightarrow$ **synth**): input matches inference (real frames), which preserves lip-audio sync; however, the target $\tilde{S}$ may contain minor artifacts from R_ϕ , so training *only* on this direction can reproduce them.
- $(\tilde{S} \rightarrow S)$ (**synth** $\rightarrow$ **real**): target is clean (real S), which discourages artifacts; but the input is synthetic and does not match inference, so training *only* on this direction hurts sync/generalization.

Mixture. We tried various training strategies, but eventually we train our model on a mixture of both directions: sample $(S \rightarrow \tilde{S})$ with probability $2/3$ and $(\tilde{S} \rightarrow S)$ with probability $1/3$. Empirically, this preserves lip-audio alignment (same LipScore as the real $\rightarrow$ synth-only variant) while improving visual quality by avoiding propagation of reconstruction artifacts. This self-refinement removes the need for external segmentation at inference and keeps the pipeline mask-free.

Losses

Let $\mathbf{M}$ be the lower-face pixel mask, and $\mathbf{m}$ its latent version downsampled with VAE. Lips mask $\mathbf{M}_{\text{lips}}$ comes from face parsing.

Let $\text{MAE}_{\mathbf{w}}(A) = \frac{1}{|\Omega_{\mathbf{w}}|} \|\mathbf{w} \odot A\|_1$ and $\text{MAE}(A) = \frac{1}{|\Omega|} \|A\|_1$, where $\Omega_{\mathbf{w}}$ indicates a support after applying the mask $\mathbf{w}$, and $\odot$ is the element-wise (Hadamard) product.

Let $\Delta \mathbf{z} = \hat{\mathbf{z}}_{\text{target}} - \mathbf{z}_{\text{target}}$ be the difference between predicted and ground-truth latents. Then, the losses in the latent space are defined as:

$$\mathcal{L}_{L1}^{\text{lat}} = \text{MAE}(\Delta \mathbf{z}), \quad \mathcal{L}_{L1_m}^{\text{lat}} = \text{MAE}_{\mathbf{m}}(\Delta \mathbf{z}). \quad (5.4)$$

We also use the following losses in the pixel space:

$$\mathcal{L}_{L1_M}^{\text{pix}} = \text{MAE}_{\mathbf{M}}(\hat{\mathbf{x}}_{\text{src}} - \mathbf{x}_{\text{src}}), \quad (5.5)$$

$$\mathcal{L}_{L1_{\text{lips}}}^{\text{pix}} = \mathbb{1}_{\{|\Omega_{\text{lips}}| \geq \tau_{\text{lips}}\}} \text{MAE}_{\mathbf{M}_{\text{lips}}}(\hat{\mathbf{x}}_{\text{src}} - \mathbf{x}_{\text{src}}), \quad (5.6)$$

$$\mathcal{L}_{VGG} = \sum_l \text{MAE}(\phi_l(\hat{\mathbf{x}}_{\text{src}}) - \phi_l(\mathbf{x}_{\text{src}})), \quad (5.7)$$

$$\mathcal{L}_{VGG}^{\text{face}} = \sum_l \text{MAE}(\psi_l(\hat{\mathbf{x}}_{\text{src}}) - \psi_l(\mathbf{x}_{\text{src}})), \quad (5.8)$$

where $\hat{\mathbf{x}}_{\text{src}}$ and $\mathbf{x}_{\text{src}}$ are predicted and ground-truth images, respectively, $\mathcal{L}_{VGG}$ uses VGG-19 features as in (Johnson *et al.*, 2016), and $\mathcal{L}_{VGG}^{\text{face}}$ uses a VGGFace2-pretrained network (Cao *et al.*, 2018). The lips loss is applied only when a valid lips mask is found and its area exceeds

τ_{lips} .

Finally, the total loss becomes:

$$\mathcal{L}_{\text{total}} = 0.1 \mathcal{L}_{L1}^{\text{lat}} + 0.1 \mathcal{L}_{L1_m}^{\text{lat}} + 10 \mathcal{L}_{L1_M}^{\text{pix}} + 100 \mathcal{L}_{L1_{\text{lips}}}^{\text{pix}} + 50 \mathcal{L}_{VGG} + 5 \mathcal{L}_{VGG}^{\text{face}} \quad (5.9)$$

5.3.2 Stage 2: Audio-to-Lips with Flow Matching

Stage 2 predicts the lips vector from speech and drives the editor trained in Stage 1. The model is a transformer conditioned on wav2vec 2.0 features (Baevski *et al.*, 2020b). We train it with a flow-matching objective (Lipman *et al.*, 2022; Liu *et al.*, 2022) in the space of lips vectors.

Let $\mathbf{a}$ be the audio features aligned to a video frame, and $\mathbf{z}_{\text{mouth}}$ the target lips vector. We follow the optimal transport conditional flow-matching. We sample $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$ and $t \sim \mathcal{U}(0, 1)$, and define an interpolated point in latent space

$$\mathbf{z}_t = (1 - t) \boldsymbol{\epsilon} + t \mathbf{z}_{\text{lips}}, \quad (5.10)$$

and the target velocity

$$\mathbf{u} = \mathbf{z}_{\text{lips}} - \boldsymbol{\epsilon}. \quad (5.11)$$

The transformer v_θ is trained to match this velocity field,

$$\mathcal{L}_{\text{FM}} = \mathbb{E}_{t, \boldsymbol{\epsilon}, \mathbf{a}} \left\| v_\theta(\mathbf{z}_t, t, \mathbf{c}) - \mathbf{u} \right\|_2^2, \quad (5.12)$$

where $\mathbf{c} = \text{Concat}[\mathbf{a}, e(\mathbf{a}), \mathbf{z}_{\text{lips}}^K]$, $e(\mathbf{a})$ is a pre-trained audio emotion encoder, and $\mathbf{z}_{\text{lips}}^K$ are K randomly sampled source lip latents.

At inference, we predict the lips pose $\hat{\mathbf{z}}_{\text{lips}}$ from audio and source lip latents, then pass it to Stage 1.

5.3.3 Architecture Details

All Stage 1 editors (Reconstruction and *LipsChange*) operate on the SDXL VAE latent grid (stride 8). The *input* is the channel-wise concatenation of the masked target latent, the identity-adapted reference latent $f_{\text{ref}}(\mathbf{z}_{\text{ref}})$, and the lips-pose vector tiled to the latent resolution; this totals **52** channels in our implementation. The network predicts a **4-channel latent residual** that is added to the masked latent and decoded by the frozen VAE. *LipsChange* shares the same backbone and is initialized from the Reconstruction network weights for mask-free self-refinement.

U-Net. Table 5.1 summarizes the U-Net model: we use ResNet2D blocks (GroupNorm (GN) 32, SiLU, 3×3 convolutions). The down path increases the number of channels from $384 \rightarrow 512 \rightarrow 640$ using average-pooling for downsampling; the up path mirrors this structure with skip concatenations and resize-conv upsampling, ending with GN+SiLU+Conv to 4 channels. This backbone yields the best throughput (see Table 5.6) while preserving identity and background consistency and confining edits to the mouth.

Stage	Composition	Channels (in → out)
Input	Conv2d($k = 3, s = 1, p = 1$)	52 → 384
Down Block 1	$4 \times \text{ResNet2D} + 1 \times \text{ResNet2D}$ (with AvgPool Downsample)	384 → 384
Down Block 2	$4 \times \text{ResNet2D} + 1 \times \text{ResNet2D}$ (with AvgPool Downsample)	384 → 512
Down Block 3	$4 \times \text{ResNet2D}$ (no downsampling)	512 → 640
Mid Block	$2 \times \text{ResNet2D}$	640 → 640
Up Block 1	$5 \times \text{ResNet2D} + 1 \times \text{ResNet2D}$ (with Upsample)	1280 → 640
Up Block 2	$5 \times \text{ResNet2D} + 1 \times \text{ResNet2D}$ (with Upsample)	1152 → 512
Up Block 3	$5 \times \text{ResNet2D}$ (no upsampling)	896 → 384
Output	GN(32) + SiLU + Conv2d($k = 3, s = 1, p = 1$)	384 → 4

Table 5.1: **Architecture of the U-Net base model.** Each ResNet2D block consists of GroupNorm (GN, 32 groups), SiLU activation, and two Conv2d layers ($k = 3, s = 1, p = 1$).

Transformer. Table 5.2 summarizes the ViT-style model: a 1×1 input projection ($52 \rightarrow 128$), followed by GN and a 1×1 lift to 1024 channels; then 16 BasicTransformerBlocks (LayerNorm, MHSA with 16 heads $\times$ 64 dim, GEGLU MLP with $4 \times$ expansion); followed by 1×1 projections back to 4 output channels. Convolutional pre-/post-projections preserve the 2D grid, while attention improves global consistency at the cost of lower FPS.

Stage	Composition	Channels (in $\rightarrow$ out)
Input Projection	Conv2d($k = 1, s = 1$)	52 $\rightarrow$ 128
Transformer Pre-projection	GroupNorm(32) + Conv2d($k = 1, s = 1$)	128 $\rightarrow$ 1024
Transformer Blocks	16 $\times$ BasicTransformerBlock: LayerNorm + MHSA (16 heads, 64-dim/head) + LayerNorm + FeedForward (GEGLU, 4 $\times$ expansion)	1024 $\rightarrow$ 1024
Transformer Post-projection	Conv2d($k = 1, s = 1$)	1024 $\rightarrow$ 128
Output Projection	Conv2d($k = 1, s = 1$)	128 $\rightarrow$ 4

Table 5.2: **Architecture of the Transformer base model.** MHSA stands for Multi-Head Self-Attention.

Trade-off. Both backbones achieve comparable accuracy. The transformer is slightly stronger on perceptual metrics, whereas the U-Net is substantially faster. This makes the U-Net preferable for real-time use and the transformer preferable for peak visual quality. The Stage-2 flow-matching transformer (FMT) architecture – shared by both U-Net and Transformer variants of FlashLips – is detailed in Table 5.3.

Stage	Composition	Dims (in $\rightarrow$ out)
Input Motion Embedding	SequenceEmbed: Linear($d_p \rightarrow d_h$) (no affine norm)	$d_p \rightarrow d_h$ 12 $\rightarrow$ 1024
Positional Encoding	Fixed sinusoidal encoding (non-learnable), added to token embeddings ($T = 60$ frames)	$T \times d_h \rightarrow T \times d_h$ $60 \times 1024 \rightarrow 60 \times 1024$
Time Embedding	TimestepEmbedder: sinusoidal (256-dim) + MLP: Linear(256 $\rightarrow d_h$) + SiLU + Linear($d_h \rightarrow d_h$)	256 $\rightarrow d_h$ 256 $\rightarrow$ 1024 1024 $\rightarrow$ 1024
Condition Embedding	Concat of identity, audio and emotion latents: [w_r, w_a, w_e] with Linear($d_{\text{cond}} \rightarrow d_h$)	($d_{\text{cond}} \rightarrow d_h$) $d_{\text{cond}} = d_p \cdot n_{\text{id}} + d_a + d_e$ ($12n_{\text{id}} + 512 + 7$) $\rightarrow$ 1024
FMT Blocks	8 $\times$ FMTBlock: AdaLN-modulated MHSA (8 heads, 128-dim/head) + AdaLN-modulated MLP MLP: Linear($d_h \rightarrow 4d_h$) + GELU + Linear($4d_h \rightarrow d_h$) AdaLN MLP: SiLU + Linear($d_h \rightarrow 6d_h$)	$d_h \rightarrow d_h$ 1024 $\rightarrow$ 1024 1024 $\rightarrow$ 4096 4096 $\rightarrow$ 1024 1024 $\rightarrow$ 6144
Output Decoder	AdaLN: LayerNorm (no affine) + SiLU + Linear($d_h \rightarrow 2d_h$) Linear($d_h \rightarrow d_w$)	$d_h \rightarrow d_h$ 1024 $\rightarrow$ 2048 1024 $\rightarrow$ 12

Table 5.3: **Architecture of the Flow Matching Transformer (FMT).** d_p is the lips-pose latent dimension, d_h is the hidden size. MHSA stands for Multi-Head Self-Attention.

Model	FID ↓	FVD ↓	HyperIQA ↑	VBench ↑	LipScore ↑	ID ↑	PSNR ↑	SSIM ↑	LPIPS ↓
Reconstruction									
DiffDub Liu <i>et al.</i> , 2024	16.11	120.24	57.91	0.673	0.37	0.58	26.65	0.89	0.114
Diff2Lip Mukhopadhyay <i>et al.</i> , 2024	8.76	57.96	68.31	0.672	0.47	0.79	25.64	0.94	0.056
TalkLip Wang <i>et al.</i> , 2023	13.97	78.66	64.52	0.667	0.57	0.79	32.03	0.94	0.062
LatentSync Li <i>et al.</i> , 2025	5.30	36.47	73.10	<u>0.682</u>	0.55	0.86	33.61	0.97	0.015
IP-LAP Zhong <i>et al.</i> , 2023	7.91	39.89	69.57	0.674	0.40	0.82	<u>32.97</u>	<u>0.95</u>	0.033
KeySync Bigata <i>et al.</i> , 2025b	5.48	24.80	74.18	0.681	0.56	0.81	30.39	0.93	0.030
FlashLips – U-Net (Ours)	<u>4.75</u>	<u>15.20</u>	73.81	0.687	<u>0.70</u>	<u>0.85</u>	32.86	0.94	0.022
FlashLips – Transformer (Ours)	4.43	12.31	<u>74.06</u>	0.687	0.71	0.86	32.88	0.94	<u>0.021</u>
Cross-Audio									
DiffDub Liu <i>et al.</i> , 2024	18.31	123.34	57.91	0.668	0.30	0.59	–	–	–
Diff2Lip Mukhopadhyay <i>et al.</i> , 2024	9.55	80.43	69.01	0.667	0.31	0.76	–	–	–
TalkLip Wang <i>et al.</i> , 2023	17.64	134.70	64.62	0.645	0.21	0.74	–	–	–
LatentSync Li <i>et al.</i> , 2025	7.69	46.08	72.52	0.680	0.33	0.84	–	–	–
IP-LAP Zhong <i>et al.</i> , 2023	9.05	43.13	69.49	0.670	0.25	<u>0.81</u>	–	–	–
KeySync Bigata <i>et al.</i> , 2025b	6.81	37.55	74.13	0.676	0.36	0.79	–	–	–
FlashLips – U-Net (Ours)	<u>6.23</u>	<u>33.57</u>	73.58	<u>0.681</u>	0.38	<u>0.81</u>	–	–	–
FlashLips – Transformer (Ours)	5.89	29.40	<u>73.84</u>	0.682	<u>0.37</u>	<u>0.81</u>	–	–	–

Table 5.4: **Quantitative comparison.** Comparison on reconstruction and cross-audio scenarios over 100 randomly sampled reconstruction videos and 100 cross-audio pairs from HDTF, CelebV-HQ, and CelebV-Text. Best results are **bold**; second-best are underlined.

5.4 Experiments

We evaluate *FlashLips* end-to-end under two standard protocols: (i) *reconstruction* (same-clip audio) and (ii) *cross-audio* (driven by a different clip). For both settings, we report lip–audio synchronization, visual quality, identity preservation, and runtime efficiency. The full set of metrics is defined in Sec. 5.4.3.

5.4.1 Datasets

A key advantage of our design is that training the most compute-heavy component (Stage 1) does *not* require audio or even lip-sync supervision: we only need images with a visible mouth. This allows us to leverage substantially broader data than typical lip-sync pipelines. The datasets used are explained in more detail in Section 2.1.2.

Stage 1 (image-only). We train the reconstruction/self-refinement editor on MEAD (Wang *et al.*, 2020), FEED (Drobyshev *et al.*, 2024), HDTF (Zhang *et al.*, 2021), NeRSemble (Kirschstein *et al.*, 2023), CelebV-Text (Yu *et al.*, 2023), CelebV-HQ (Zhu *et al.*, 2022), and our internal talking face dataset.

Stage 2 (audio-visual). For audio-to-lips, we use Hallo3 (Cui *et al.*, 2024), RAVDESS (Liv-

Model	Reconstruction						
	SC $\uparrow$	BC $\uparrow$	MS $\uparrow$	DD $\uparrow$	AQ $\uparrow$	IQ $\uparrow$	Total $\uparrow$
DiffDub Liu <i>et al.</i> , 2024	<u>0.962</u>	0.954	0.992	0.670	0.505	0.661	0.673
Diff2Lip Mukhopadhyay <i>et al.</i> , 2024	0.953	0.946	0.992	0.653	0.557	0.633	0.672
TalkLip Wang <i>et al.</i> , 2023	0.952	0.942	0.992	<u>0.727</u>	0.528	0.596	0.667
LatentSync Li <i>et al.</i> , 2025	0.967	0.948	<u>0.991</u>	0.723	0.528	0.671	<u>0.682</u>
IP_LAP Zhong <i>et al.</i> , 2023	0.961	0.945	0.992	0.720	0.519	0.640	0.674
KeySync Bigata <i>et al.</i> , 2025b	0.953	0.948	<u>0.991</u>	0.750	0.531	<u>0.669</u>	0.681
FlashLips – U-Net (Ours)	0.957	0.956	0.990	0.750	<u>0.559</u>	0.667	0.687
FlashLips – Transformer (Ours)	0.957	<u>0.955</u>	0.990	0.750	0.560	<u>0.669</u>	0.687

Model	Cross-Audio						
	SC $\uparrow$	BC $\uparrow$	MS $\uparrow$	DD $\uparrow$	AQ $\uparrow$	IQ $\uparrow$	Total $\uparrow$
DiffDub Liu <i>et al.</i> , 2024	0.956	0.953	0.992	0.624	0.506	0.660	0.668
Diff2Lip Mukhopadhyay <i>et al.</i> , 2024	0.946	0.945	<u>0.991</u>	0.622	0.550	0.631	0.667
TalkLip Wang <i>et al.</i> , 2023	<u>0.958</u>	0.947	0.992	0.420	0.527	0.596	0.645
LatentSync Li <i>et al.</i> , 2025	0.963	0.953	<u>0.991</u>	0.690	0.537	0.664	0.680
IP_LAP Zhong <i>et al.</i> , 2023	0.963	0.945	0.992	0.670	0.519	0.639	0.670
KeySync Bigata <i>et al.</i> , 2025b	0.951	0.949	<u>0.991</u>	<u>0.680</u>	0.529	0.668	0.676
FlashLips – U-Net (Ours)	0.955	0.958	0.990	0.670	<u>0.558</u>	<u>0.666</u>	<u>0.681</u>
FlashLips – Transformer (Ours)	0.955	<u>0.957</u>	0.990	<u>0.680</u>	0.559	0.668	0.682

Table 5.5: **Quantitative comparison on VBench.** Video quality evaluation using VBench Huang *et al.*, 2023 metrics on 100 randomly sampled reconstruction videos and 100 cross-audio pairs from HDTF, CelebV-HQ and CelebV-Text. Metrics defined in Section 5.4.3.

ingstone and Russo, 2018), MEAD, HDTF, NeRSemble, CelebV-Text, CelebV-HQ, and our private dataset. Stage 2 is trained to predict the lips vector from aligned speech features.

For quantitative evaluation, we randomly sample 100 reconstruction videos and 100 cross-audio pairs from HDTF, CelebV-HQ and CelebV-Text eval sets.

5.4.2 Training Details

Data Augmentation During training for both stages, we apply the following augmentations. All images are normalized by dividing by 255 to map pixel values into the range $[0, 1]$. For Stage 1, we additionally apply random horizontal flips with probability 0.5 and ColorJitter with a coefficient of 0.4 for hue, brightness, and contrast, 3.2 for saturation, and an overall application probability of 0.9. For the distilled mouth-latent network, we downscale the source image to 384×384 using interpolation.

Model	FPS $\uparrow$	Speedup $\times$
DiffDub Liu <i>et al.</i> , 2024	1.86	58.8
Diff2Lip Mukhopadhyay <i>et al.</i> , 2024	19.77	5.5
TalkLip Wang <i>et al.</i> , 2023	51.53	2.1
LatentSync Li <i>et al.</i> , 2025	5.70	19.2
IP_LAP Zhong <i>et al.</i> , 2023	4.24	25.8
KeySync Bigata <i>et al.</i> , 2025b	3.60	30.4
FlashLips – U-Net (Ours)	109.41	1.0
FlashLips – Transformer (Ours)	<u>66.84</u>	1.6

Table 5.6: **Inference speed.** Speed comparison in frames per second (FPS). “Speedup” denotes the inference speed gain of our fastest model (FlashLips – U-Net) over each method. Measured on the same clip: 5 warm-ups, then 10 runs to average FPS.

Stage 1 (Reconstruction & LipsChange). For the one-step editor, we have two backbone versions: a U-Net with 250M parameters and a ViT-like visual transformer with 300M. Both are optimized with AdamW and a OneCycleLR schedule: the learning rate starts at 1×10^{-4} and anneals to 1×10^{-8} . We train the reconstruction phase for 1 million iterations on $8 \times$ NVIDIA H100 80GB HBM3 GPUs with a total batch size of 32. After convergence, we train the mask-free self-refinement *LipsChange* (Sec. 5.3.1) using symmetric *source* $\leftrightarrow$ *changed* pseudo-pairs for 200k iterations.

Lips Encoder and distillation. As shown in Fig. 5.4, our *Lips Encoder* starts from a frozen expression encoder (Drobyshev *et al.*, 2024) that maps a face crop to an expression latent, followed by a small trainable head (ResNet-50 + MLP) that projects to the compact lips-pose vector used by Stage 1. We train this head during the Stage 1 reconstruction phase. To remove pre-processing overhead at inference (the expression backbone requires several alignment/cropping steps), we *distill* the pipeline into a single ResNet-50 that consumes the full image and predicts the same lips-pose vector. Distillation is performed by matching the outputs of the full pipeline over the last 200k iterations of Stage 1 reconstruction training.

Stage 2 (Audio-to-Lips). Stage 2 is a transformer conditioned on wav2vec 2.0 features, emotion class inferred from audio, and K random lip latents and trained with a conditional flow-matching objective in the lips-pose vector space (Sec. 5.3.2). We use the same lips-pose vector definition as Stage 1 to ensure compatibility. For training, we utilize the AdamW

Model	# Ref Lats	FID ↓	FVD ↓	HyperIQA ↑	LipScore ↑	ID ↑	PSNR ↑	SSIM ↑	LPIPS ↓
Reconstruction									
Transformer with Mask	1	8.06	57.80	73.26	0.50	0.81	27.68	0.90	0.044
	4	8.01	57.93	73.27	0.53	0.82	27.75	0.90	0.043
	8	7.95	58.15	73.28	0.55	0.82	27.77	0.90	0.043
	16	7.97	57.78	73.29	0.56	0.82	27.79	0.90	0.043
	32	7.94	57.63	73.28	0.55	0.82	27.80	0.90	0.043
Transformer Mask-free	1	4.46	12.53	74.05	0.69	0.85	32.71	0.94	0.021
	4	4.43	12.31	74.06	0.71	0.86	32.88	0.94	0.021
	8	4.41	12.47	74.07	0.73	0.85	33.00	0.94	0.021
	16	4.38	11.90	74.08	0.74	0.86	33.02	0.94	0.021
	32	4.36	12.16	74.08	0.75	0.86	33.10	0.94	0.020
U-Net Mask-free	1	4.73	15.68	73.79	0.67	0.85	32.74	0.94	0.022
	4	4.75	15.20	73.81	0.70	0.85	32.86	0.94	0.022
	8	4.76	15.61	73.82	0.70	0.85	32.92	0.94	0.022
	16	4.66	15.07	73.83	0.71	0.85	32.95	0.94	0.022
	32	4.70	15.85	73.83	0.69	0.85	32.97	0.94	0.022
Cross-Audio									
Transformer with Mask	1	10.24	73.91	73.16	0.40	0.77	—	—	—
	4	9.87	68.10	73.23	0.39	0.77	—	—	—
	8	9.74	68.77	73.26	0.38	0.78	—	—	—
	16	9.67	66.17	73.25	0.35	0.79	—	—	—
	32	9.68	64.92	73.25	0.34	0.78	—	—	—
Transformer Mask-free	1	6.25	41.38	73.80	0.40	0.79	—	—	—
	4	5.89	29.40	73.84	0.37	0.81	—	—	—
	8	5.81	29.54	73.88	0.35	0.81	—	—	—
	16	5.73	26.35	73.88	0.34	0.81	—	—	—
	32	5.68	25.16	73.89	0.32	0.82	—	—	—
U-Net Mask-free	1	6.54	42.78	73.51	0.40	0.79	—	—	—
	4	6.23	33.57	73.58	0.38	0.81	—	—	—
	8	6.13	32.64	73.61	0.36	0.80	—	—	—
	16	6.07	31.34	73.63	0.34	0.81	—	—	—
	32	6.13	28.54	73.63	0.32	0.81	—	—	—

Table 5.7: **Full ablation study.** Ablation study of our mask and mask-free models, and different numbers of references for the audio-to-latent model for reconstruction and cross-audio. Metrics computed on 100 randomly sampled reconstruction videos and 100 cross-audio pairs from HDTF, CelebV-HQ and CelebV-Text.

optimizer with default parameters and a constant 5×10^{-5} learning rate. The transformer backbone has 150M parameters.

5.4.3 Evaluation Metrics

We evaluate models using a diverse set of metrics that target lip–audio synchronization, visual quality, identity preservation, and efficiency.

To assess lip–audio synchronization accuracy, we use LipScore (Bigata *et al.*, 2025a). For reconstruction fidelity and perceptual quality, we adopt both reference-based and no-reference metrics: Fréchet Inception Distance (FID) (Heusel *et al.*, 2017) at the frame level and Fréchet

Lips Encoder Variant	PSNR $\uparrow$	SSIM $\uparrow$	LipScore $\uparrow$	ID $\uparrow$
	Reconstruction		Cross-Audio	
V1 2D	27.11	0.88	0.30	0.83
V1 4D	30.74	0.91	0.32	0.83
V1 8D	31.63	0.93	0.34	0.83
V1 16D	31.95	0.93	0.34	<u>0.82</u>
V1 8D + V2 2D	32.16	0.93	0.36	0.81
V1 8D + V2 4D	32.86	0.94	<u>0.38</u>	0.80
V1 8D + V2 8D	<u>34.86</u>	<u>0.95</u>	<u>0.38</u>	0.74
V1 8D + V2 16D	36.86	0.97	0.40	0.63

Table 5.8: **Lips encoder ablation.** V1 (frozen expression encoder) saturates near 8D; adding a lips-crop residual (V2) improves reconstruction but reduces cross-audio ID. The 12D setting (V1 8D + V2 4D) offers the best quality–disentanglement trade-off.

Video Distance (FVD) (Unterthiner *et al.*, 2019b) at the video level for overall visual quality and temporal consistency; LPIPS (Zhang *et al.*, 2018) as a reference-based perceptual similarity measure; and HyperIQA (Su *et al.*, 2020) as a no-reference image-quality metric correlated with human perception. We also report Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM) (Wang *et al.*, 2004) for pixel-level structural similarity.

To evaluate identity preservation, we define an identity (ID) metric as the cosine similarity between predicted and ground-truth face embeddings computed per frame using FaceNet-512 (Schroff *et al.*, 2015; Serengil and Ozpinar, 2024). For overall visual consistency, we report VBench (Huang *et al.*, 2023) scores: Subject Consistency (SC), Background Consistency (BC), Motion Smoothness (MS), Dynamic Degree (DD), Aesthetic Quality (AQ), Imaging Quality (IQ), and an aggregated “Total” - the mean of the normalized metrics.

Finally, to measure efficiency, we report inference speed in frames per second (FPS). For each model, the same clip is processed 5 times for warm-up and 10 times for measurement, and we report the average runtime over the measured runs.

5.4.4 Quantitative Evaluation

We evaluate reconstruction and cross-audio performance against recent state-of-the-art methods, including DiffDub (Liu *et al.*, 2024), Diff2Lip (Mukhopadhyay *et al.*, 2024), TalkLip (Wang

et al., 2023), LatentSync (Li *et al.*, 2025), IP-LAP (Zhong *et al.*, 2023), and KeySync (Bigata *et al.*, 2025b). Results are shown in Tables 5.4 and 5.6, with full VBench metrics in Table 5.5.

Both our FlashLips models achieve the best overall FID and FVD in both protocols, while also being substantially faster than the next-best method (up to **30.4** $\times$ vs. KeySync). On Hyper-IQA, we rank second and third with only marginal differences from the top model, a metric commonly used as a proxy for human perceptual quality, and obtain the highest VBench total score, indicating strong perceptual quality, temporal consistency, and background preservation.

In LipScore, our models are the top two in both protocols, ensuring highest lip-sync quality. In identity preservation (ID), we are first in reconstruction tied with LatentSync, and second in cross-audio behind it, tied with IP-LAP, while being up to **19.2** $\times$ and **25.8** $\times$ faster, respectively. This reflects strong lip-audio alignment and identity retention across the full sequence.

For pixel-level metrics (PSNR, SSIM), we place third with a narrow gap, and second in LPIPS, showing that our one-step reconstruction maintains high image fidelity.

Overall, FlashLips delivers state-of-the-art visual and synchronization quality while maintaining faster-than-real-time throughput: **109.4** FPS for the U-Net and **66.8** FPS for the Transformer base models.

5.4.5 Qualitative Evaluation

Qualitative results in Fig. 5.7 and Fig. 5.8 show that both FlashLips variants produce visually realistic, temporally coherent animations with strong identity preservation. Compared to existing approaches, our method yields smoother and more natural mouth motion without introducing artifacts or over-exaggerated animation. LatentSync and IP-LAP maintain identity reasonably well but often show weak lip-audio synchronization and noticeable visual artifacts; Diff2Lip exhibits similar issues. DiffDub and Wav2Lip exhibit identity drift and inconsistent lip motion, while KeySync struggles with large head poses, causing deformation and temporal instability. In contrast, FlashLips remains robust across more diverse viewing

angles (Fig. 5.9) and maintains stable lip–audio alignment. It also shows good generalization to out-of-distribution and non-human identities (Fig. 5.10).

5.4.6 User Study

To complement our quantitative evaluation, we conducted a user study comparing FlashLips with several baseline lip-sync models. Using the same 100 cross-audio videos as in our quantitative experiments, we present participants with two videos per trial: one generated by our method and one by a randomly selected baseline. Users evaluate either *Visual Quality* or *Lip Sync*, choosing the preferred video or indicating that both are of the same quality. We collect up to 700 votes per baseline comparison and setting, which are then aggregated. The results are shown in Figure 5.5.

Across nearly all baselines, FlashLips is the clear user preference for both visual quality and lip-sync accuracy, with a substantial portion of responses also indicating comparable quality. We outperform DiffDub, Diff2Lip, TalkLip, and IP_LAP by a large margin, with only a minority of votes favoring the competing models. Against LatentSync, most users judge the outputs to be similar, with a slight preference for our method. KeySync – a considerably slower (by $\times 30.4$ times, see Table 5.6) iterative diffusion model – shows a negligible advantage with 29.0% vs 32.7% of votes for visual quality and 26.6% vs 28.4% for lip-sync, although the vast majority of users still deem the two videos to be of equal quality with 38.3% and 45.0% of votes in the respective settings. We attribute this small disadvantage to artifacts introduced by the SDXL VAE under certain framings and head poses (see Section 5.6).

Overall, the study highlights that FlashLips delivers competitive or superior perceptual quality while operating orders of magnitude faster than state-of-the-art diffusion-based approaches.

5.4.7 Mask Removal: Quantitative Impact

To isolate the effect of removing explicit mouth masks, Table 5.7 compares the *Transformer with Mask* to our mask-free editors *Transformer Mask-free* and *U-Net Mask-free* under identical

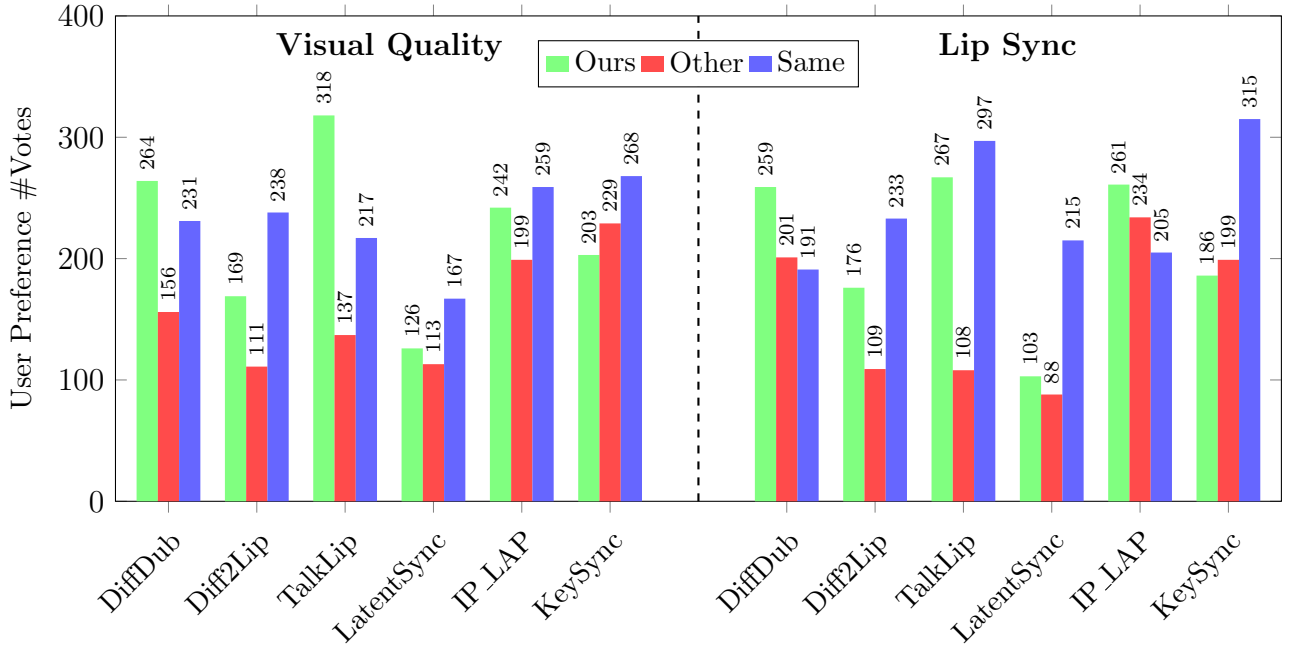


Figure 5.5: **Human preference evaluation.** We conducted a user study comparing FlashLips against randomly selected baseline models. Participants indicated their preference across two criteria: Visual Quality and Lip Sync. The chart displays the number of responses favoring our model (Ours), the competing model (Other), or neither (Same).

evaluation protocols. We treat reconstruction as a sanity check and focus primarily on cross-audio, which reflects the real use case of our model.

Reconstruction. Removing the mask improves both fidelity and lip-sync quality for the Transformer variant: LipScore increases from roughly 0.50 to 0.70–0.75, and all fidelity metrics (FID/FVD, LPIPS, PSNR, SSIM, ID) move in the expected direction, suggesting better distribution match, sharper frames and higher identity similarity. This confirms that mask-free editing can localize mouth modifications without sacrificing reconstruction quality.

Cross-audio. Mask removal yields the largest improvements in cross-audio. For the Transformer, FID drops from ~ 10.2 to ~ 5.7 and FVD from ~ 74 to ~ 25 – 41 , indicating cleaner frames and substantially more stable motion. ID improves from ~ 0.77 – 0.79 to ~ 0.81 – 0.82 , and Hyper-IQA increases slightly. LipScore remains in the same range of 0.35–0.40, showing that lip–audio alignment is preserved. Qualitatively, mask-free models reduce mouth glitches and flicker while providing more stable backgrounds and facial detail. The mask-free U-Net follows the same

trend, with slightly worse FID/FVD but higher throughput.

Takeaway. Mask-free self-refinement is a key contributor to the final system: it removes the need for segmentation at inference and consistently improves perceptual quality, temporal smoothness, and identity preservation, while maintaining lip-audio alignment comparable to or better than the masked baseline.

5.5 Ablation Study

Lips Encoder design and dimensionality. Our goal is a lips vector that (a) captures *configuration* of the lips and jaw and is easy to predict from audio, while (b) carrying minimal appearance information (skin tone, lip/teeth geometry) that should instead come from the reference identity latent. Lower appearance leakage simplifies Stage 2 (which cannot infer appearance from audio) and stabilizes training. It also improves pseudo-pair generation for self-refinement: with a well-disentangled vector, we can borrow lips-poses across identities to increase pose diversity.

We experiment with two principal variants. **(V1)** Use only the frozen expression encoder from (Drobyshev *et al.*, 2024) followed by a small projection head. **(V2)** Augment *V1* with a small *residual* predicted from a lips crop (extracted via face parsing) by a lightweight ResNet. Table 5.8 shows that the **(V1)** variant already performs strongly with low identity leakage; reconstruction gains saturate around a lips vector of size 8. Adding a residual progressively improves reconstruction, but hurts disentanglement, especially after the residual reaches size 8. To test leakage, we substituted lips-pose vectors from different identities. We therefore adopt a **12D** mouth vector with an 8D encoder component + 4D residual, which offers a good trade-off: high reconstruction quality and low identity leakage.

Number of reference frames. We vary the number of reference lips-pose vectors used in Stage 2. As shown in Table 5.7, moving from 1 to 4 references improves identity preservation

with negligible impact on sync; beyond 4 gains are marginal and we occasionally observe sync degradation – likely because too much identity information reduces the work done by audio-to-lips network and increases sensitivity when references come from a different clip at inference. We therefore use **4 reference frames**.

Stage 1 Backbone: U-Net vs. Transformer. Both backbones have similar accuracy (Table 5.4): the transformer is slightly better perceptually but slower, while the U-Net is faster. We use the U-Net for speed and the transformer for higher visual quality.

Summary. The ablations support three design choices used in the main results: a 12D lip-pose vector (8D encoder + 4D residual), four reference frames, and either a U-Net (speed) or transformer (quality) backbone for Stage 1.

5.6 Limitations

Although our model produces high-quality lip-sync in most cases, it still exhibits some limitations (Figure 5.6). Since the method relies on direct prediction rather than the iterative denoising used in diffusion-based approaches, it can struggle to generate fine-grained facial details, particularly in regions such as facial hair and teeth. While the model was not explicitly trained to handle occlusions, it is often surprisingly robust; however, occluding objects can still degrade lip-sync accuracy in more challenging sequences. A more fundamental limitation stems from the SDXL VAE, whose performance degrades in a predictable manner under certain framings and head poses. The VAE performs well on tight close-ups, but when the subject appears in wider shots, artifacts become more common and can adversely affect the lip-sync quality.

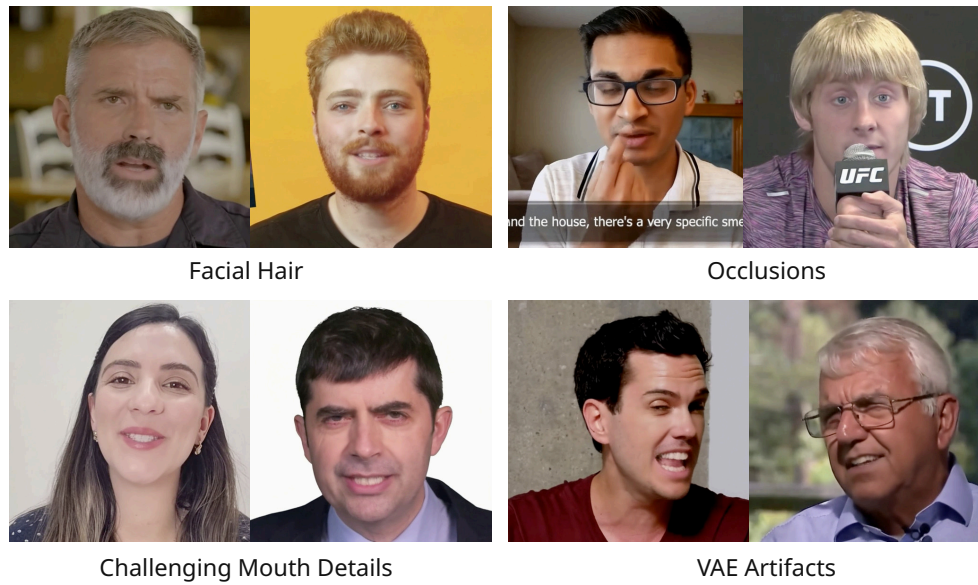


Figure 5.6: **Limitations.** Examples illustrating typical failure cases under challenging conditions, including generating facial hair and teeth details, occlusions, and artifacts caused by the SDXL VAE.

5.7 Conclusion

FlashLips shows that lip-sync can be posed as *deterministic editing*: a one-step, reconstruction-only latent editor driven by a flow-matched audio-to-lips module, with self-refinement removing the need of masks at inference. This yields a simple, modular system that runs faster than real time (>100 FPS with the U-Net variant) while matching recent diffusion/GAN baselines in visual quality, identity retention, and lip-audio alignment. Our analysis indicates that a compact, disentangled control vector plus self-refinement are sufficient to localize edits to the mouth, avoid identity/background drift, and decouple *what* to render from *how* to render, making the approach practical for dubbing and other production settings. Future work includes improving robustness under occlusions and extreme motion, and enriching the control space with prosody- and emotion-aware signals.

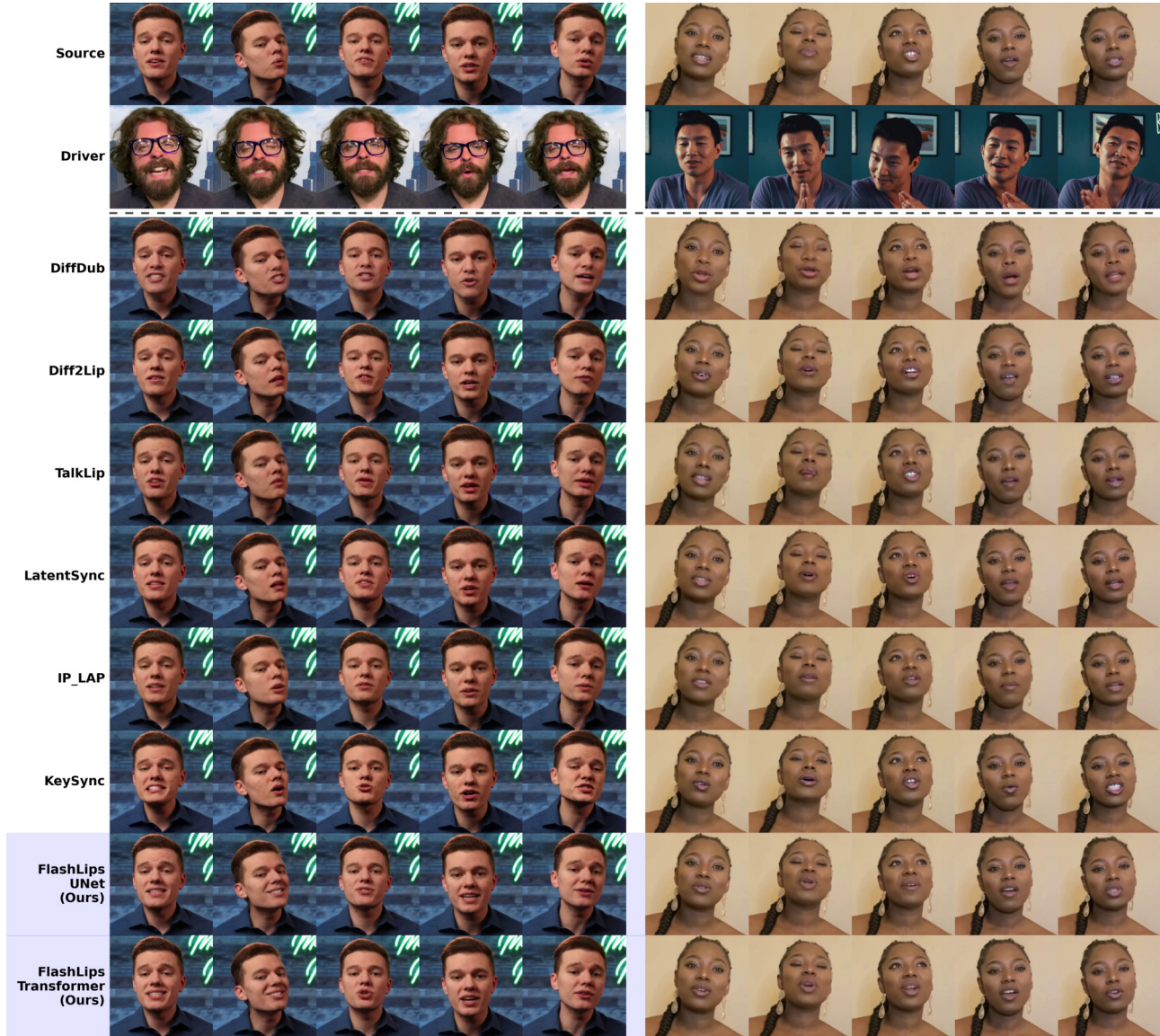


Figure 5.7: **Qualitative comparison – cross-audio.** Comparison with other lip-sync methods for cross-audio. The first row shows the source video, the second row shows the video corresponding to the audio driver. The following rows display the inferred lip-synced videos by each method.



Figure 5.8: **Qualitative comparison – reconstruction.** Comparisons with other lip-sync methods for reconstruction. The first row shows the source video; the following rows display the inferred lip-synced videos by each method.

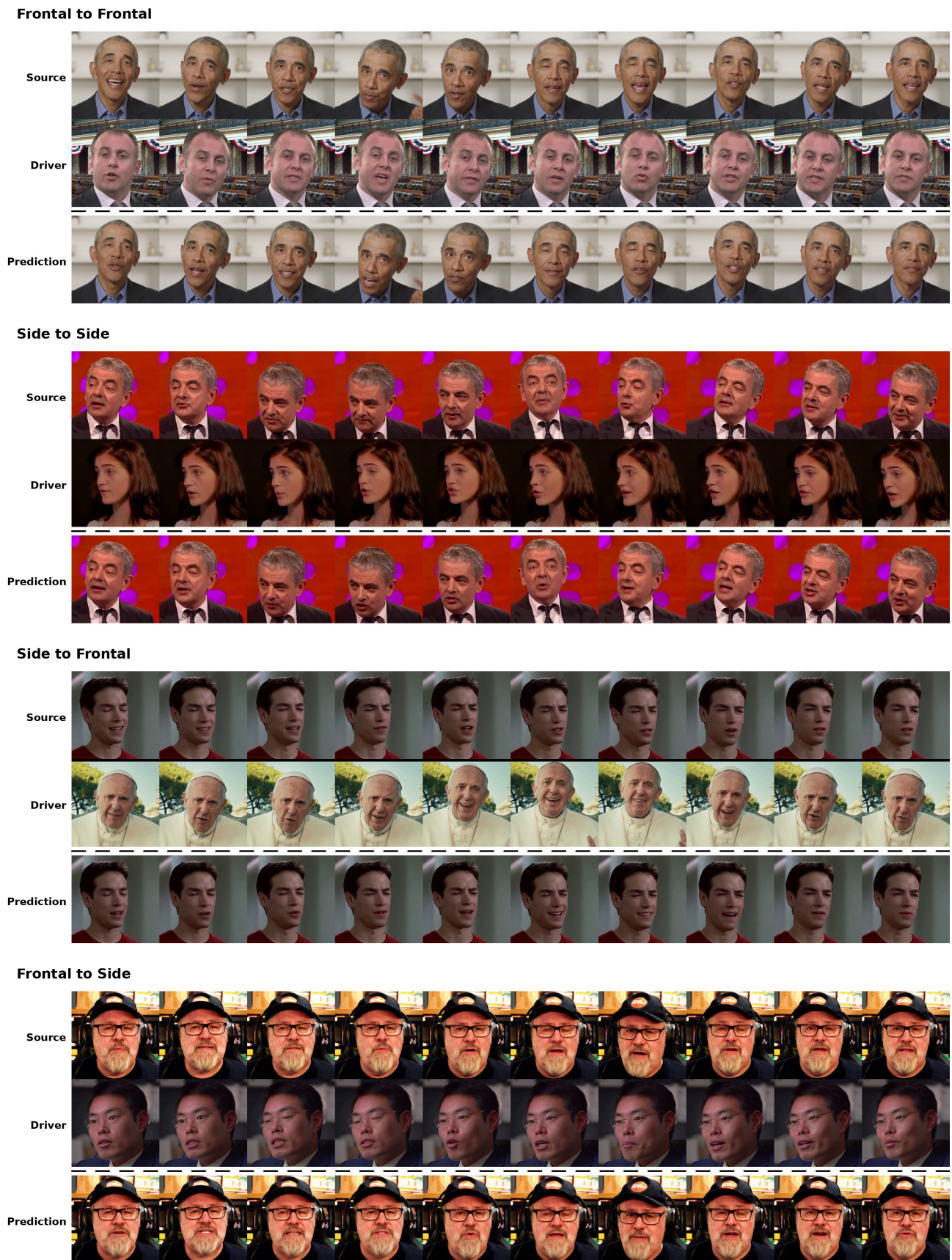


Figure 5.9: Lip-sync results across varying facial pose combinations. Each triplet shows a source video, video corresponding to the audio driver, and the resulting prediction.



Figure 5.10: **Lip-sync results on out-of-distribution (OOD) faces.** The top block of images shows results on generated human faces, while the lower block shows results on non-human or stylized faces. Our method maintains consistent lip synchronization and natural articulation across both domains.

CHAPTER 6

CONCLUSION

Contents

6.1	Summary	116
6.2	Ethical Considerations	118
6.2.1	Speech Recognition	118
6.2.2	Lip Synchronization	119
6.3	Future Work	120

6.1 Summary

This thesis focuses on the problem of learning useful representations from audio and video by using deep learning and leveraging large amounts of unlabelled data. We generally focus our efforts on two tasks: speech recognition and lip synchronization. Below, we summarize our contributions from each chapter.

In Chapter 3, we investigated whether self-supervised learning can be extended to multilingual settings for visual speech recognition. Using the self-supervised RAVeN framework for audio-visual pre-training, we evaluated how different combinations of languages affect downstream lip-reading performance on video. Our experiments showed that multilingual pre-training improves CER when sufficient data is available, and that these gains cannot be attributed solely to increased dataset size – non-English languages contribute complementary information. We also found that linguistic similarity affects transfer quality, and that fine-tuning on languages unseen during pre-training still yields competitive results. Overall, this chapter demonstrates that cross-lingual self-supervision is both feasible and beneficial for visual speech representation learning.

In Chapter 4, we introduced BRAVeN, an improved audio-visual self-supervised framework built on RAVeN that addresses the semantic imbalance between audio and visual speech cues. By smoothing teacher targets, simplifying the video predictor, applying stronger audio masking, and re-weighting audio losses, BRAVeN produces more informative representations for both modalities. We showed that these design choices lead to consistent gains over existing self-supervised approaches, and that BRAVeN scales effectively with both model size and the quantity of unlabelled data. Using up to 3,000 hours of raw audio-visual speech and only 30 hours of labelled data, BRAVeN achieved state-of-the-art performance in both VSR and ASR, demonstrating that high-quality audio-visual recognition is possible with minimal annotation.

In Chapter 5, we presented FlashLips, a fast and modular lip synchronization framework that challenges the need for adversarial or diffusion-based generators in this highly conditioned task. By separating control from rendering, we introduced a lightweight one-step latent editor trained purely with reconstruction losses, followed by a mask-free self-refinement stage that sharpens lip edits without requiring external segmentation. We then linked audio to this editor through a disentangled audio-to-pose transformer trained with flow matching, ensuring that pose information is predicted from speech while appearance remains fixed. Together, these components enable faster-than-real-time performance at over 100 FPS, while achieving competitive or superior lip-sync accuracy compared to much heavier iterative models. This chapter demonstrates that deterministic, one-step generation is both feasible and effective for lip-sync, offering a

simpler and faster alternative to current diffusion-based approaches.

Overall, the contributions of this thesis demonstrate the effectiveness and versatility of deep representation learning across a range of audio-visual tasks. In speech recognition, we show that large-scale unlabelled data can be effectively leveraged within a self-supervised framework to learn powerful latent representations, requiring only limited supervision during fine-tuning for the downstream task. For lip synchronization, we again operate entirely in latent space, employing reconstruction-based objectives and visual or audio inputs to construct a fully label-free generative pipeline. Across these settings, a consistent finding is that audio and visual speech exhibit rich, naturally aligned structure that can be exploited as a form of mutual supervision. Leveraging this cross-modal synchrony enables models to acquire information that would otherwise require manual annotation, thereby improving robustness, scalability, and generalization across audio-visual tasks. More broadly, this thesis highlights how representation learning can reduce dependence on labelled data, simplify model design, and enable efficient, scalable audio-visual learning.

6.2 Ethical Considerations

6.2.1 Speech Recognition

Audio-visual speech recognition offers several positive applications, particularly in accessibility. It can support deaf or hard-of-hearing users, assist with speech-to-text in noisy environments, and facilitate communication for individuals unable to use a keyboard or touch-screen. Such systems can also improve customer service workflows by enabling faster responses, accurate transcriptions, and automated analysis of user interactions, thereby reducing staff workload and improving service quality.

At the same time, these technologies introduce important risks. Speech recognition may be misused for surveillance or non-consensual transcription, enabling eavesdropping from silent videos or the recovery of spoken content without a speaker’s awareness. Any real-world deploy-

ment must therefore be transparent and require informed consent from affected individuals. Additional privacy concerns arise from data storage and processing; for instance, audio-visual data uploaded to cloud servers may be vulnerable to leaks, and devices that passively listen for trigger words may inadvertently capture private conversations. Errors in transcription can also lead to serious consequences in high-stakes settings such as healthcare or legal documentation. Furthermore, due to the limited availability of diverse ASR and VSR datasets, current models may exhibit demographic or linguistic biases, resulting in uneven performance across languages, accents, facial characteristics, or recording conditions. These risks highlight the need for responsible development, continuous monitoring, and caution when deploying speech-recognition systems in sensitive environments.

6.2.2 Lip Synchronization

Lip-sync technology enables a wide range of useful applications. It supports seamless dubbing for film and television, allowing translated content to retain the emotional nuance of the original performance. It can also power expressive avatars in video games and virtual environments, making interactions more natural and engaging. Moreover, lip-sync automation offers substantial time and cost savings in content creation by reducing the need for manual dubbing or re-filming when the content needs to be updated.

However, these capabilities also come with significant risks. Malicious actors may use lip-sync systems to generate deceptive deepfakes, spread misinformation, or impersonate others. Such misuse could lead to political manipulation, identity theft, or social engineering attacks that trick individuals into revealing sensitive information. Harm may also arise from unauthorised impersonations – especially of public figures – which can cause reputational damage or psychological distress. Fabricated audio-visual evidence, such as falsified alibis or misleading video clips, poses additional threats to legal and institutional processes. Our method is intended solely for beneficial applications, and we explicitly discourage any harmful or non-consensual use. Any system that modifies a person’s likeness should require explicit, informed consent. Finally, as with many audio-visual models, dataset limitations can introduce biases across at-

tributes such as skin tone, facial structure, language, or accent, necessitating careful evaluation prior to deployment.

6.3 Future Work

The field of deep learning is rapidly evolving, offering numerous directions to extend the work presented in this thesis. Below, we outline several promising research directions.

A possible research direction relates to our work in Chapter 3. In this chapter, we examined how different multilingual audio-visual pre-training configurations affect visual-only fine-tuning. A natural extension is to explore joint audio-visual fine-tuning, which may exploit complementary phonetic and articulatory information more effectively than visual fine-tuning alone. Applying our newer BRAVEN architecture in the multilingual setting also remains unexplored and could yield improved cross-lingual generalization. Another direction concerns data augmentation for low-resource languages. Synthetic audio generated using modern text-to-speech (TTS) models (Radford *et al.*, 2022; contributors, 2024) or synthetic video from portrait-animation systems (Bigata *et al.*, 2025a) could supplement scarce training data, and prior work has shown that combining real and synthetic data can be beneficial (Liu *et al.*, 2023; Feng *et al.*, 2024).

Another direction comes from our work in Chapter 4. In this chapter, we introduced BRAVEN as an improved self-supervised architecture for audio-visual speech recognition. Future work could explore alternative training regimes and targets to learn representations that transfer to other tasks, such as emotion recognition, speaker identity, or controlling generative models for audio-to-video or video-to-audio synthesis. Large Language Models (LLMs) (Minaee *et al.*, 2025) also present an interesting opportunity: multimodal LLMs (Cappellazzo *et al.*, 2025; Yin *et al.*, 2024) could provide linguistic priors or high-level semantic constraints during ASR/VSR/AVSR fine-tuning, potentially improving robustness. Finally, our model currently uses separate audio and visual encoders, which increases model complexity and parameter count. Recent work on unified multimodal encoders (Haliassos *et al.*, 2024a) suggests that a shared architecture may reduce redundancy and improve cross-modal consistency, making this a promising direction for

future exploration.

Further possible directions arise from our work in Chapter 5. A core limitation of our current model is the underlying auto-encoder, whose latent bottleneck can degrade the quality of the output and introduces artifacts. We would therefore like to experiment with more powerful auto-encoders (Labs *et al.*, 2025; Kong *et al.*, 2024) that better preserve high-frequency details in latent space, as well as with recent approaches that operate directly and efficiently in pixel space to avoid quality loss (Chen *et al.*, 2025a). To further improve fidelity, we could explore a hybrid solution that uses direct reconstruction for the overall image structure and diffusion or flow-matching models for refining high-frequency regions such as teeth and facial hair. We could also investigate the use of LLMs as additional guidance, for example by conditioning on a description of an emotion, a description of the subject, or a transcription of the audio. Finally, our current approach assumes a single visible subject; extending it to handle two or more subjects in the frame, and thus perform lip-sync in multi-speaker or group situations (Wang *et al.*, 2025), would broaden its applicability to real conversational scenarios.

BIBLIOGRAPHY

- [1] P. Ma, S. Petridis, and M. Pantic, “End-to-end audio-visual speech recognition with conformers,” in *ICASSP*, 2021, pp. 7613–7617.
- [2] T. Afouras, J. S. Chung, A. Senior, O. Vinyals, and A. Zisserman, “Deep audio-visual speech recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.
- [3] A. Nagrani, J. S. Chung, and A. Zisserman, “Voxceleb: A large-scale speaker identification dataset,” *Interspeech*, 2017.
- [4] Y. Bengio, A. Courville, and P. Vincent, *Representation learning: A review and new perspectives*, 2014. arXiv: 1206.5538 [cs.LG]. [Online]. Available: <https://arxiv.org/abs/1206.5538>.
- [5] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.
- [6] Y. LeCun, Y. Bengio, *et al.*, “Convolutional networks for images, speech, and time series,” *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.
- [7] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

- [8] A. Vaswani *et al.*, “Attention is all you need,” *Advances in neural information processing systems*, vol. 30, 2017.
- [9] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” in *International Conference on Learning Representations (ICLR)*, 2017.
- [10] A. Dosovitskiy *et al.*, “An image is worth 16x16 words: Transformers for image recognition at scale,” in *International Conference on Learning Representations (ICLR)*, 2021.
- [11] R. Balestrieri, J. Zarka, A. Ramesh, *et al.*, “A cookbook of self-supervised learning,” *Proceedings of the IEEE*, 2023.
- [12] A. v. d. Oord, Y. Li, and O. Vinyals, “Representation learning with contrastive predictive coding,” in *arXiv preprint arXiv:1807.03748*, 2018.
- [13] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, “Momentum contrast for unsupervised visual representation learning,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9729–9738, 2020.
- [14] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, “Wav2vec 2.0: A framework for self-supervised learning of speech representations,” *arXiv preprint arXiv:2006.11477*, 2020.
- [15] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, “A simple framework for contrastive learning of visual representations,” in *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020, pp. 1597–1607.
- [16] J.-B. Grill *et al.*, “Bootstrap your own latent: A new approach to self-supervised learning,” in *Advances in neural information processing systems*, vol. 33, 2020, pp. 21 271–21 284.
- [17] M. Caron *et al.*, “Emerging properties in self-supervised vision transformers,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 9630–9640.
- [18] R. Arandjelovic and A. Zisserman, “Look, listen and learn,” in *Proceedings of the 16th IEEE/CVF International Conference on Computer Vision (ICCV)*, 2017, pp. 609–617.

- [19] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, “Masked autoencoders are scalable vision learners,” *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16 000–16 009, 2022.
- [20] W.-N. Hsu, B. Bolte, Y.-H. H. Tsai, K. Lakhotia, R. Salakhutdinov, and A. Mohamed, “Hubert: Self-supervised speech representation learning by masked prediction of hidden units,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 3451–3460, 2021.
- [21] B. Shi, W. Hsu, K. Lakhotia, and A. Mohamed, “Learning audio-visual speech representation by masked multimodal cluster prediction,” in *ICLR*, 2022. [Online]. Available: <https://openreview.net/forum?id=Z1Qlm11u0M>.
- [22] H. Zhou, Y. Chen, J. Cao, Y. Zhang, W. Wang, and Z. Liu, “Talking face generation by adversarially disentangled audio-visual representation,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019.
- [23] K. R. Prajwal, R. Mukhopadhyay, V. P. Namboodiri, and C. Jawahar, “A lip sync expert is all you need for speech to lip generation in the wild,” in *Proceedings of the 28th ACM International Conference on Multimedia*, ser. MM ’20, ACM, Oct. 2020, 484–492. DOI: 10.1145/3394171.3413532. [Online]. Available: <http://dx.doi.org/10.1145/3394171.3413532>.
- [24] A. Bigata *et al.*, *Keyface: Expressive audio-driven facial animation for long sequences via keyframe interpolation*, 2025. arXiv: 2503.01715 [cs.CV]. [Online]. Available: <https://arxiv.org/abs/2503.01715>.
- [25] A. Bigata *et al.*, *Keysync: A robust approach for leakage-free lip synchronization in high resolution*, 2025. arXiv: 2505.00497 [cs.CV]. [Online]. Available: <https://arxiv.org/abs/2505.00497>.
- [26] H. Zhou *et al.*, “Makeittalk: Speaker-aware talking head animation,” in *ACM Transactions on Graphics (TOG)*, 2020.
- [27] Y. Wang *et al.*, “Mead: A large-scale audio-visual dataset for emotional talking-face generation,” in *ACM International Conference on Multimedia*, 2020.

- [28] Y. Lipman, N. Param, J. Ho, R. Salakhutdinov, and Y. Carmon, “Flow matching for generative modeling,” *arXiv preprint arXiv:2205.14809*, 2022.
- [29] J. Ho, A. Jain, and P. Abbeel, “Denoising diffusion probabilistic models,” in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, ser. NIPS ’20, Vancouver, BC, Canada: Curran Associates Inc., 2020, ISBN: 9781713829546.
- [30] A. Zinonos, A. Haliassos, P. Ma, S. Petridis, and M. Pantic, “Learning cross-lingual visual speech representations,” *arXiv preprint arXiv:2303.09455*, 2023.
- [31] A. Haliassos, P. Ma, R. Mira, S. Petridis, and M. Pantic, *Jointly learning visual and auditory speech representations from raw data*, 2023. arXiv: 2212.06246 [cs.LG]. [Online]. Available: <https://arxiv.org/abs/2212.06246>.
- [32] A. Haliassos, A. Zinonos, R. Mira, S. Petridis, and M. Pantic, “Braven: Improving self-supervised pre-training for visual and auditory speech recognition,” in *ICASSP*, 2024.
- [33] A. Zinonos, M. Stypulkowski, A. Bigata, S. Petridis, M. Pantic, and N. Drobyshev, *Flashlips: 100-fps mask-free latent lip-sync using reconstruction instead of diffusion organs*, 2025. arXiv: 2512.20033 [cs.CV]. [Online]. Available: <https://arxiv.org/abs/2512.20033>.
- [34] A. Ephrat *et al.*, “Looking to listen at the cocktail party: A speaker-independent audio-visual model for speech separation,” *ACM Trans. Graph.*, vol. 37, no. 4, p. 112, 2018. DOI: 10.1145/3197517.3201357. [Online]. Available: <https://doi.org/10.1145/3197517.3201357>.
- [35] J. Valk and T. Alumäe, “Voxlingua107: A dataset for spoken language recognition,” in *2021 IEEE Spoken Language Technology Workshop (SLT)*, 2021, pp. 652–658. DOI: 10.1109/SLT48900.2021.9383459.
- [36] J. S. Chung, A. Nagrani, and A. Zisserman, “Voxceleb2: Deep speaker recognition,” in *Interspeech*, 2018, pp. 1086–1090.
- [37] T. Afouras, J. S. Chung, and A. Zisserman, “LRS3-TED: A large-scale dataset for visual speech recognition,” *arXiv*, 2018.

- [38] J. S. Chung, A. Senior, O. Vinyals, and A. Zisserman, “Lip reading sentences in the wild,” in *Proceedings of the 30th IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 3444–3453.
- [39] Y. Zhao, R. Xu, and M. Song, “A cascade sequence-to-sequence model for chinese mandarin lip reading,” in *Proc. MMAsia*, 2019, 32:1–32:6. DOI: 10.1145/3338533.3366579. [Online]. Available: <https://doi.org/10.1145/3338533.3366579>.
- [40] E. Salesky *et al.*, “The multilingual tedx corpus for speech recognition and translation,” in *Proc. Interspeech*, 2021, pp. 3655–3659. DOI: 10.21437/Interspeech.2021-11. [Online]. Available: <https://doi.org/10.21437/Interspeech.2021-11>.
- [41] Z. Zhang, L. Li, Y. Ding, and C. Fan, “Flow-guided one-shot talking face generation with a high-resolution audio-visual dataset,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 3661–3670.
- [42] N. Drobyshev, A. B. Casademunt, K. Vougioukas, Z. Landgraf, S. Petridis, and M. Pantic, *Emoportraits: Emotion-enhanced multimodal one-shot head avatars*, 2024. arXiv: 2404.19110 [cs.CV].
- [43] T. Kirschstein, S. Qian, S. Giebenhain, T. Walter, and M. Nießner, “Nersemble: Multi-view radiance field reconstruction of human heads,” *ACM Transactions on Graphics (TOG)*, vol. 42, no. 4, pp. 1–14, 2023.
- [44] J. Yu, H. Zhu, L. Jiang, C. C. Loy, W. Cai, and W. Wu, “CelebV-Text: A large-scale facial text-video dataset,” in *CVPR*, 2023.
- [45] H. Zhu *et al.*, “CelebV-HQ: A large-scale video facial attributes dataset,” in *ECCV*, 2022.
- [46] S. R. Livingstone and F. A. Russo, “The ryerson audio-visual database of emotional speech and song (ravdess): A dynamic, multimodal set of facial and vocal expressions in north american english,” *PloS one*, vol. 13, no. 5, e0196391, 2018.
- [47] J. Cui *et al.*, *Hallo3: Highly dynamic and realistic portrait image animation with video diffusion transformer*, 2024. arXiv: 2412.00733 [cs.CV].

- [48] PyTorch Team, *Audio feature extractions tutorial*, https://pytorch.org/audio/main/tutorials/audio_feature_extractions_tutorial.html, Accessed: 2025-07-20, 2024.
- [49] P. Ma, S. Petridis, and M. Pantic, “Visual speech recognition for multiple languages in the wild,” *Nature Machine Intelligence*, vol. 4, no. 11, pp. 930–939, 2022.
- [50] J. Deng, J. Guo, E. Ververas, I. Kotsia, and S. Zafeiriou, “Retinaface: Single-shot multi-level face localisation in the wild,” in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 5202–5211. DOI: 10.1109/CVPR42600.2020.00525.
- [51] A. Bulat and G. Tzimiropoulos, “How far are we from solving the 2d & 3d face alignment problem?(and a dataset of 230,000 3d facial landmarks),” in *Proceedings of the 16th IEEE/CVF International Conference on Computer Vision (ICCV)*, 2017, pp. 1021–1030.
- [52] S. Su *et al.*, “Blindly assess image quality in the wild guided by a self-adaptive hyper network,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2020.
- [53] B. Castellano, *PySceneDetect*, 2014. [Online]. Available: <https://github.com/Breakthrough/PySceneDetect>.
- [54] J. Liao, H. Duan, K. Feng, W. Zhao, Y. Yang, and L. Chen, “A light weight model for active speaker detection,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2023, pp. 22 932–22 941.
- [55] T. Kudo and J. Richardson, “Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing,” in *ACL*, 2018, pp. 66–71.
- [56] A. Karpathy, J. Johnson, and F.-F. Li, *Cs231n convolutional neural networks for visual recognition: Neural networks part 1*, <https://cs231n.github.io/neural-networks-1/>, Accessed: 2025-07-22, 2016.
- [57] A. F. Agarap, “Deep learning using rectified linear units (relu),” *arXiv preprint arXiv:1803.08375*, 2018.

- [58] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *Nature*, vol. 323, no. 6088, pp. 533–536, 1986. DOI: 10.1038/323533a0.
- [59] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, *Efficient BackProp*. Springer, 1998, pp. 9–50. DOI: 10.1007/3-540-49430-8_2.
- [60] D. V. Godoy, *Dl-visuals: Deep learning visuals*, <https://github.com/dvgodoy/dl-visuals>, GitHub repository. License: CC-BY-4.0. Accessed: 2025-07-22, 2021.
- [61] V. Dumoulin and F. Visin, *A guide to convolution arithmetic for deep learning*, 2018. arXiv: 1603.07285 [stat.ML]. [Online]. Available: <https://arxiv.org/abs/1603.07285>.
- [62] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, ser. JMLR Workshop and Conference Proceedings, vol. 37, 2015, pp. 448–456.
- [63] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.
- [64] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2015, pp. 234–241.
- [65] J. L. Elman, “Finding structure in time,” *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.
- [66] K. Cho *et al.*, “Learning phrase representations using rnn encoder-decoder for statistical machine translation,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1724–1734.
- [67] J. L. Ba, J. R. Kiros, and G. E. Hinton, “Layer normalization,” *arXiv preprint arXiv:1607.06450*, 2016.

- [68] A. Graves, A.-r. Mohamed, and G. Hinton, “Speech recognition with deep recurrent neural networks,” in *2013 IEEE international conference on acoustics, speech and signal processing*, IEEE, 2013, pp. 6645–6649.
- [69] S. Petridis, T. Stafylakis, P. Ma, and et al., “End-to-end audiovisual speech recognition,” *ICASSP*, 2018.
- [70] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, “X-vectors: Robust dnn embeddings for speaker recognition,” in *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, IEEE, 2018, pp. 5329–5333.
- [71] T. Afouras, J. Chung, and A. Zisserman, “Deep lip reading: A comparison of models and an online application,” *interspeech*, 2022.
- [72] L. Goncalves, S.-G. Leem, W.-C. Lin, B. Sisman, and C. Busso, *Versatile audio-visual learning for emotion recognition*, 2024. arXiv: 2305.07216 [cs.LG]. [Online]. Available: <https://arxiv.org/abs/2305.07216>.
- [73] Y. Bengio, A. Courville, and P. Vincent, “Representation learning: A review and new perspectives,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.
- [74] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng, “Multimodal deep learning,” in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 2011, pp. 689–696.
- [75] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, “High-resolution image synthesis with latent diffusion models,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 10 684–10 695.
- [76] Lawtomed. “Supervised vs unsupervised learning: Which is better?” Accessed: 2024-06-18. (2021), [Online]. Available: <https://lawtomed.com/supervised-vs-unsupervised-learning-which-is-better/>.
- [77] L. Jing and Y. Tian, “Self-supervised visual feature learning with deep neural networks: A survey,” *IEEE transactions on pattern analysis and machine intelligence*, 2020.

- [78] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.
- [79] S. Ren, K. He, R. Girshick, and J. Sun, *Faster r-cnn: Towards real-time object detection with region proposal networks*, 2016. arXiv: 1506.01497 [cs.CV]. [Online]. Available: <https://arxiv.org/abs/1506.01497>.
- [80] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., vol. 25, Curran Associates, Inc., 2012. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.
- [81] R. Balestrieri, J. Zarka, A. Baevski, and et al., “A cookbook of self-supervised learning,” *Proceedings of the IEEE*, 2023, <https://arxiv.org/abs/2304.12210>.
- [82] J. Devlin, M. Chang, K. Lee, and K. Toutanova, “BERT: pre-training of deep bidirectional transformers for language understanding,” in *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT*, 2019, pp. 4171–4186.
- [83] M. Caron, P. Bojanowski, A. Joulin, and M. Douze, “Deep clustering for unsupervised learning of visual features,” in *ECCV*, 2018.
- [84] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin, “Unsupervised learning of visual features by contrasting cluster assignments,” in *NeurIPS*, 2020.
- [85] A. Baevski, W.-N. Hsu, Q. Xu, and et al., “Data2vec: A general framework for self-supervised learning in speech, vision and language,” *arXiv preprint arXiv:2202.03555*, 2022.
- [86] H. Alwassel and et al., “Self-supervised learning by cross-modal audio-video clustering,” *arXiv preprint arXiv:1911.12667*, 2020.

- [87] P. Ma, R. Mira, S. Petridis, B. W. Schuller, and M. Pantic, “LiRA: Learning visual speech representations from audio through self-supervision,” in *Interspeech*, 2021, pp. 3011–3015.
- [88] G. E. Hinton and R. R. Salakhutdinov, “Reducing the dimensionality of data with neural networks,” *Science*, vol. 313, no. 5786, pp. 504–507, 2006. DOI: 10.1126/science.1127647.
- [89] D. Bank, N. Koenigstein, and R. Giryes, “Autoencoders,” *CoRR*, vol. abs/2003.05991, 2020. arXiv: 2003.05991. [Online]. Available: <https://arxiv.org/abs/2003.05991>.
- [90] L. Weng, “From autoencoder to Beta-vae,” *lilianweng.github.io*, Aug. 2018. [Online]. Available: <https://lilianweng.github.io/posts/2018-08-12-vae/>.
- [91] D. P. Kingma and M. Welling, “Auto-Encoding Variational Bayes,” in *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. arXiv: <http://arxiv.org/abs/1312.6114v10> [stat.ML].
- [92] A. van den Oord, O. Vinyals, and K. Kavukcuoglu, “Neural discrete representation learning,” in *Advances in Neural Information Processing Systems*, vol. 30, 2017.
- [93] I. Goodfellow *et al.*, “Generative adversarial nets,” in *Advances in neural information processing systems*, vol. 27, 2014.
- [94] Buff ML. “Generative adversarial network (gan).” Accessed: 2025-08-02. (2022), [Online]. Available: <https://buffml.com/generative-adversarial-network-gan/>.
- [95] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet: A Large-Scale Hierarchical Image Database,” in *CVPR09*, 2009.
- [96] K. Vougioukas, S. Petridis, and M. Pantic, “Realistic speech-driven facial animation with gans,” *CoRR*, vol. abs/1906.06337, 2019. arXiv: 1906.06337. [Online]. Available: <http://arxiv.org/abs/1906.06337>.
- [97] S. Suwajanakorn, S. M. Seitz, and I. Kemelmacher-Shlizerman, “Synthesizing obama: Learning lip sync from audio,” in *ACM Transactions on Graphics (TOG)*, vol. 36, 2017, pp. 1–13.

- [98] L. Chen *et al.*, “Hierarchical cross-modal talking face generation with dynamic pixel-wise loss,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 7832–7841.
- [99] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro, “Video-to-video synthesis,” in *Advances in neural information processing systems*, vol. 31, 2018.
- [100] M. Arjovsky, S. Chintala, and L. Bottou, “Wasserstein gan,” in *Proceedings of the International Conference on Machine Learning*, 2017.
- [101] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, “Spectral normalization for generative adversarial networks,” in *International Conference on Learning Representations*, 2018.
- [102] S. Tulyakov, M.-Y. Liu, X. Yang, and J. Kautz, “Mocogan: Decomposing motion and content for video generation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1526–1535.
- [103] J. Ho, A. Jain, and P. Abbeel, “Denoising diffusion probabilistic models,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 6840–6851, 2020.
- [104] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, *Score-based generative modeling through stochastic differential equations*, 2021. arXiv: 2011.13456 [cs.LG]. [Online]. Available: <https://arxiv.org/abs/2011.13456>.
- [105] T. Karras, T. Aila, S. Laine, and J. Lehtinen, “Progressive growing of gans for improved quality, stability, and variation,” in *International Conference on Learning Representations (ICLR)*, 2018.
- [106] A. Krizhevsky, “Learning multiple layers of features from tiny images,” University of Toronto, Tech. Rep., 2009, Technical report.
- [107] J. Ho and T. Salimans, “Classifier-free diffusion guidance,” in *NeurIPS Workshop on Deep Generative Models and Downstream Applications*, 2022.
- [108] P. A. Mittal and et al., “Avatardiff: Talking face generation via latent diffusion models,” in *CVPR*, 2023.

- [109] C. Li *et al.*, *Latentsync: Taming audio-conditioned latent diffusion models for lip sync with syncnet supervision*, 2025. arXiv: 2412.09262 [cs.CV]. [Online]. Available: <https://arxiv.org/abs/2412.09262>.
- [110] V. Popov, I. Vovk, A. Polyak, and et al., “Grad-tts: A diffusion probabilistic model for text-to-speech,” *arXiv preprint arXiv:2105.06337*, 2021.
- [111] D. J. Rezende and S. Mohamed, “Variational inference with normalizing flows,” in *International Conference on Machine Learning (ICML)*, 2015, pp. 1530–1538.
- [112] L. Dinh, J. Sohl-Dickstein, and S. Bengio, “Density estimation using real nvp,” in *International Conference on Learning Representations (ICLR)*, 2017.
- [113] J. Cho, Y. Lee, J. Yoo, *et al.*, “Diffusion-free talking face generation with flow matching,” *arXiv preprint arXiv:2309.15333*, 2023.
- [114] H. Wang, X. Liu, *et al.*, “Flow-guided audio-visual generation,” *arXiv preprint arXiv:2312.11548*, 2024.
- [115] F. Du *et al.*, *Rap: Real-time audio-driven portrait animation with video diffusion transformer*, 2025. arXiv: 2508.05115 [cs.GR]. [Online]. Available: <https://arxiv.org/abs/2508.05115>.
- [116] I. Loshchilov and F. Hutter, “Decoupled weight decay regularization,” in *ICLR*, 2019.
- [117] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.
- [118] I. Loshchilov and F. Hutter, “Sgdr: Stochastic gradient descent with warm restarts,” in *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, 2017.
- [119] L. N. Smith and N. Topin, “Super-convergence: Very fast training of neural networks using large learning rates. arxiv,” *arXiv preprint arXiv:1708.07120*, vol. 6, 2017.
- [120] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, “Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks,” vol. 2006, Jan. 2006, pp. 369–376. DOI: 10.1145/1143844.1143891.

- [121] R. J. Williams and D. Zipser, “A learning algorithm for continually running fully recurrent neural networks,” *Neural computation*, vol. 1, no. 2, pp. 270–280, 1989.
- [122] S. Kim, T. Hori, and S. Watanabe, “Joint ctc-attention based end-to-end speech recognition using multi-task learning,” in *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, IEEE, 2017, pp. 4835–4839.
- [123] M. Freitag and Y. Al-Onaizan, “Beam search strategies for neural machine translation,” *arXiv preprint arXiv:1702.01806*, 2017.
- [124] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.
- [125] J. Johnson, A. Alahi, and L. Fei-Fei, “Perceptual losses for real-time style transfer and super-resolution,” in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds., Cham: Springer International Publishing, 2016, pp. 694–711, ISBN: 978-3-319-46475-6.
- [126] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, “Gans trained by a two time-scale update rule converge to a local nash equilibrium,” in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS’17, Long Beach, California, USA: Curran Associates Inc., 2017, 6629–6640, ISBN: 9781510860964.
- [127] T. Unterthiner, M. Heusel, B. Nessler, G. Klambauer, and S. Hochreiter, “Fvd: A new metric for video generation,” in *Workshop on Deep Generative Models for Highly Structured Data, ICML*, 2019.
- [128] Z. Huang *et al.*, *Vbench: Comprehensive benchmark suite for video generative models*, 2023. arXiv: 2311.17982 [cs.CV]. [Online]. Available: <https://arxiv.org/abs/2311.17982>.
- [129] P. Ma, A. Haliassos, A. Fernandez-Lopez, H. Chen, S. Petridis, and M. Pantic, “Auto-avsr: Audio-visual speech recognition with automatic labels,” in *ICASSP*, IEEE, 2023, pp. 1–5.

- [130] F. Schroff, D. Kalenichenko, and J. Philbin, “Facenet: A unified embedding for face recognition and clustering,” in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2015, 815–823. DOI: 10.1109/cvpr.2015.7298682. [Online]. Available: <http://dx.doi.org/10.1109/CVPR.2015.7298682>.
- [131] S. Serengil and A. Ozpinar, “A benchmark of facial recognition pipelines and co-usability performances of modules,” *Journal of Information Technologies*, vol. 17, no. 2, pp. 95–107, 2024. DOI: 10.17671/gazibtd.1399077. [Online]. Available: <https://dergipark.org.tr/en/pub/gazibtd/issue/84331/1399077>.
- [132] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, “The unreasonable effectiveness of deep features as a perceptual metric,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.
- [133] L. Burget *et al.*, “Multilingual acoustic modeling for speech recognition based on subspace gaussian mixture models,” in *ICASSP*, 2010, pp. 4334–4337.
- [134] G. Heigold *et al.*, “Multilingual acoustic models using distributed deep neural networks,” in *ICASSP*, 2013, pp. 8619–8623.
- [135] A. Kannan *et al.*, “Large-scale multilingual speech recognition with a streaming end-to-end model,” in *Proc. Interspeech*, 2019, pp. 2130–2134. DOI: 10.21437/Interspeech.2019-2858. [Online]. Available: <https://doi.org/10.21437/Interspeech.2019-2858>.
- [136] A. Haliassos, R. Mira, S. Petridis, and M. Pantic, “Leveraging real talking faces via self-supervision for robust forgery detection,” in *Proceedings of the 35th IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 14 950–14 962.
- [137] A. Baevski, S. Schneider, and M. Auli, “Vq-wav2vec: Self-supervised learning of discrete speech representations,” in *ICLR*, 2020. [Online]. Available: <https://openreview.net/forum?id=rylwJxrYDS>.

- [138] A. Baevski, M. Auli, and A. Mohamed, “Effectiveness of self-supervised pre-training for speech recognition,” *CoRR*, vol. abs/1911.03912, 2019. arXiv: 1911.03912. [Online]. Available: <http://arxiv.org/abs/1911.03912>.
- [139] H. Hu *et al.*, “Redat: Accent-invariant representation for end-to-end asr by domain adversarial training with relabeling,” in *Proc. ICASSP*, 2021, pp. 6408–6412.
- [140] B. Li *et al.*, “Multi-dialect speech recognition with a single sequence-to-sequence model,” in *Proc. ICASSP*, 2018, pp. 4749–4753.
- [141] P. Ma, R. Mira, S. Petridis, B. W. Schuller, and M. Pantic, “Lira: Learning visual speech representations from audio through self-supervision,” in *Proc. Interspeech*, 2021, pp. 3011–3015. DOI: 10.21437/Interspeech.2021-1360. [Online]. Available: <https://doi.org/10.21437/Interspeech.2021-1360>.
- [142] A. Conneau, A. Baevski, R. Collobert, A. Mohamed, and M. Auli, “Unsupervised cross-lingual representation learning for speech recognition,” in *Proc. Interspeech*, 2021, pp. 2426–2430. DOI: 10.21437/Interspeech.2021-329. [Online]. Available: <https://doi.org/10.21437/Interspeech.2021-329>.
- [143] A. Babu *et al.*, “Xls-r: Self-supervised cross-lingual speech representation learning at scale,” in *Proc. Interspeech*, 2022, pp. 2278–2282. DOI: 10.21437/Interspeech.2022-143. [Online]. Available: <https://doi.org/10.21437/Interspeech.2022-143>.
- [144] T. Stafylakis and G. Tzimiropoulos, “Combining residual networks with LSTMs for lipreading,” in *Interspeech*, vol. 9, 2017, pp. 3652–3656.
- [145] S. Watanabe, T. Hori, S. Kim, J. R. Hershey, and T. Hayashi, “Hybrid ctc/attention architecture for end-to-end speech recognition,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 8, pp. 1240–1253, 2017.
- [146] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning, “ELECTRA: pre-training text encoders as discriminators rather than generators,” in *Proceedings of the 8th International Conference on Learning Representations (ICLR)*, 2020.
- [147] T. Makino *et al.*, “Recurrent neural network transducer for audio-visual speech recognition,” in *ASRU Workshop*, 2019, pp. 905–912.

- [148] K. R. Prajwal, T. Afouras, and A. Zisserman, “Sub-word level lip reading with visual attention,” in *CVPR*, 2022.
- [149] D. Serdyuk, O. Braga, and O. Siohan, “Audio-visual speech recognition is worth 32x32x8 voxels,” in *ASRU Workshop*, 2021.
- [150] D. Serdyuk, O. Braga, and O. Siohan, “Transformer-based video front-ends for audio-visual speech recognition for single and multi-person video,” in *Proc. Interspeech*, 2022, pp. 2833–2837.
- [151] T. Afouras, J. S. Chung, and A. Zisserman, “Asr is all you need: Cross-modal distillation for lip reading,” in *ICASSP*, 2020, pp. 2143–2147.
- [152] X. Liu *et al.*, “Synthvsr: Scaling up visual speech recognition with synthetic supervision,” in *CVPR*, 2023, pp. 18 806–18 815.
- [153] Q. Zhu *et al.*, “Vatlm: Visual-audio-text pre-training with unified masked prediction for speech representation learning,” *IEEE Trans. Multimed.*, 2023.
- [154] M. Ravanelli *et al.*, “Multi-task self-supervised learning for robust speech recognition,” in *ICASSP*, 2020, pp. 6989–6993.
- [155] J. Lian, A. Baevski, W.-N. Hsu, and M. Auli, “Av-data2vec: Self-supervised learning of audio-visual speech representations with contextualized target representations,” *arXiv*, 2023.
- [156] D. Ulyanov, A. Vedaldi, and V. Lempitsky, “Instance normalization: The missing ingredient for fast stylization,” *arXiv preprint arXiv:1607.08022*, 2016.
- [157] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, “Deep networks with stochastic depth,” in *Proceedings of the 14th European Conference on Computer Vision (ECCV)*, 2016, pp. 646–661.
- [158] B. Shillingford *et al.*, “Large-scale visual speech recognition,” in *Interspeech*, 2019, pp. 4135–4139.
- [159] W.-N. Hsu and B. Shi, “U-hubert: Unified mixed-modal speech pretraining and zero-shot transfer to unlabeled modality,” *NeurIPS*, vol. 35, pp. 21 157–21 170, 2022.

- [160] A. Varga and H. J. Steeneken, “Assessment for automatic speech recognition: Ii. noisex-92: A database and an experiment to study the effect of additive noise on speech recognition systems,” *Speech communication*, vol. 12, no. 3, pp. 247–251, 1993.
- [161] Z. Zhang, Z. Hu, W. Deng, C. Fan, T. Lv, and Y. Ding, “Dinet: Deformation inpainting network for realistic face visually dubbing on high resolution video,” in *AAAI Conference on Artificial Intelligence*, 2023. [Online]. Available: <https://api.semanticscholar.org/CorpusID:257378103>.
- [162] R. Zhen, W. Song, Q. He, J. Cao, L. Shi, and J. Luo, “Human-computer interaction system: A survey of talking-head generation,” *Electronics*, vol. 12, no. 1, 2023, ISSN: 2079-9292. DOI: 10.3390/electronics12010218. [Online]. Available: <https://www.mdpi.com/2079-9292/12/1/218>.
- [163] M. Xu *et al.*, *Hallo: Hierarchical audio-driven visual synthesis for portrait image animation*, 2024. arXiv: 2406.08801 [cs.CV]. [Online]. Available: <https://arxiv.org/abs/2406.08801>.
- [164] Z. Chen, J. Cao, Z. Chen, Y. Li, and C. Ma, “Echomimic: Lifelike audio-driven portrait animations through editable landmark conditions,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, pp. 2403–2410, Apr. 2025. DOI: 10.1609/aaai.v39i3.32241.
- [165] C. Wang *et al.*, *V-express: Conditional dropout for progressive training of portrait video generation*, 2024. arXiv: 2406.02511 [cs.CV]. [Online]. Available: <https://arxiv.org/abs/2406.02511>.
- [166] K. R. Prajwal, R. Mukhopadhyay, V. P. Namboodiri, and C. Jawahar, “A lip sync expert is all you need for speech to lip generation in the wild,” in *Proceedings of the 28th ACM International Conference on Multimedia*, ACM, Oct. 2020, 484–492. DOI: 10.1145/3394171.3413532.
- [167] J. Guan *et al.*, “Stylesync: High-fidelity generalized and personalized lip sync in style-based generator,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2023, pp. 1505–1515.

- [168] T. Salimans *et al.*, “Improved techniques for training gans,” in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., vol. 29, Curran Associates, Inc., 2016. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2016/file/8a3363abe792db2d8761d6403605aeb7-Paper.pdf.
- [169] A. Srivastava, L. Valkov, C. Russell, M. U. Gutmann, and C. Sutton, “Veegan: Reducing mode collapse in gans using implicit variational learning,” in *Advances in Neural Information Processing Systems*, I. Guyon *et al.*, Eds., vol. 30, Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/44a2e0804995faf8d2e3b084a1e2db1d-Paper.pdf.
- [170] J. Shen, Y. Zhou, Z. Liu, J. Wang, and J. Wang, “Difftalk: A diffusion model for realistic talking head generation,” in *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.
- [171] M. Stypulkowski, K. Vougioukas, S. He, M. Zieba, S. Petridis, and M. Pantic, “Diffused heads: Diffusion models beat gans on talking-face generation,” in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, Jan. 2024, pp. 5091–5100.
- [172] Y. Zhang, Z. Liu, J. Yan, and C. Li, “Dreamtalk: When expressive 3d talking head generation meets diffusion probabilistic models,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, 2024, pp. 5555–5563.
- [173] D. Bigioi *et al.*, “Speech driven video editing via an audio-conditioned diffusion model,” *Image and Vision Computing*, vol. 142, p. 104911, 2024, ISSN: 0262-8856. DOI: <https://doi.org/10.1016/j.imavis.2024.104911>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0262885624000143>.
- [174] T. Liu, C. Du, S. Fan, F. Chen, and K. Yu, “Diffdub: Person-generic visual dubbing using inpainting renderer with diffusion auto-encoder,” in *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2024, pp. 3630–3634. DOI: 10.1109/ICASSP48485.2024.10446049.

- [175] S. Mukhopadhyay, S. Suri, R. T. Gadde, and A. Shrivastava, “Diff2lip: Audio conditioned diffusion models for lip-synchronization,” in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, Jan. 2024, pp. 5292–5302.
- [176] D. Podell *et al.*, *Sdxl: Improving latent diffusion models for high-resolution image synthesis*, 2023. arXiv: 2307.01952 [cs.CV]. [Online]. Available: <https://arxiv.org/abs/2307.01952>.
- [177] Z. Peng *et al.*, *Omnisync: Towards universal lip synchronization via diffusion transformers*, 2025. arXiv: 2505.21448 [cs.CV]. [Online]. Available: <https://arxiv.org/abs/2505.21448>.
- [178] X. Liu, C. Gong, and Q. Liu, “Flow straight and fast: Learning to generate and transfer data with rectified flow,” *arXiv preprint arXiv:2209.03003*, 2022.
- [179] L. Tian, Q. Wang, B. Zhang, and L. Bo, “Emo: Emote portrait alive generating expressive portrait videos with audio2video diffusion model under weak conditions,” in *Computer Vision – ECCV 2024*, A. Leonardis, E. Ricci, S. Roth, O. Russakovsky, T. Sattler, and G. Varol, Eds., Cham: Springer Nature Switzerland, 2025, pp. 244–260.
- [180] X. Ji *et al.*, “Sonic: Shifting focus to global audio perception in portrait animation,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2025, pp. 193–203.
- [181] J. Jiang, C. Liang, J. Yang, G. Lin, T. Zhong, and Y. Zheng, *Loopy: Taming audio-driven portrait avatar with long-term motion dependency*, 2025. arXiv: 2409.02634 [cs.CV]. [Online]. Available: <https://arxiv.org/abs/2409.02634>.
- [182] K. Vougioukas, S. Petridis, and M. Pantic, “End-to-end speech-driven facial animation with temporal gans,” in *British Machine Vision Conference*, 2018. [Online]. Available: <https://api.semanticscholar.org/CorpusID:44061183>.
- [183] H. Zhou, Y. Sun, W. Wu, C. C. Loy, X. Wang, and Z. Liu, “Pose-controllable talking face generation by implicitly modularized audio-visual representation,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2021, pp. 4176–4186.

- [184] W. Zhang *et al.*, “Sadtalker: Learning realistic 3d motion coefficients for stylized audio-driven single image talking face animation,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2023, pp. 8652–8661.
- [185] H. Wei, Z. Yang, and Z. Wang, *Aniportrait: Audio-driven synthesis of photorealistic portrait animation*, 2024. arXiv: 2403.17694 [cs.CV]. [Online]. Available: <https://arxiv.org/abs/2403.17694>.
- [186] C. Zhang *et al.*, *Dream-talk: Diffusion-based realistic emotional audio-driven method for single image talking face generation*, 2023. arXiv: 2312.13578 [cs.CV]. [Online]. Available: <https://arxiv.org/abs/2312.13578>.
- [187] S. Xu *et al.*, “Vasa-1: Lifelike audio-driven talking faces generated in real time,” *Advances in Neural Information Processing Systems*, vol. 37, pp. 660–684, 2024.
- [188] T. Ki, D. Min, and G. Chae, “Float: Generative motion latent flow matching for audio-driven talking portrait,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025, pp. 14 699–14 710.
- [189] J. S. Chung and A. Zisserman, “Out of time: Automated lip sync in the wild,” in *Proceedings of the 13th Asian Conference on Computer Vision (ACCV)*, 2016, pp. 251–263.
- [190] W. Zhong *et al.*, “Identity-preserving talking face generation with landmark and appearance priors,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2023, pp. 9729–9738.
- [191] J. Guan *et al.*, “Resyncer: Rewiring style-based generator for unified audio-visually synced facial performer,” in *Computer Vision – ECCV 2024*, A. Leonardis, E. Ricci, S. Roth, O. Russakovsky, T. Sattler, and G. Varol, Eds., Cham: Springer Nature Switzerland, 2025, pp. 348–367.
- [192] T. Ki and D. Min, “Stylelipsync: Style-based personalized lip-sync video generation,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct. 2023, pp. 22 841–22 850.

- [193] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, “Analyzing and improving the image quality of stylegan,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2020.
- [194] J. Wang, X. Qian, M. Zhang, R. T. Tan, and H. Li, “Seeing what you said: Talking face generation guided by a lip reading expert,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 14 653–14 662.
- [195] J. Ma *et al.*, *Sayanything: Audio-driven lip synchronization with conditional video diffusion*, 2025. arXiv: 2502.11515 [cs.CV]. [Online]. Available: <https://arxiv.org/abs/2502.11515>.
- [196] Y. Zhang *et al.*, *Musetalk: Real-time high-fidelity video dubbing via spatio-temporal sampling*, 2025. arXiv: 2410.10122 [cs.CV]. [Online]. Available: <https://arxiv.org/abs/2410.10122>.
- [197] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman, *Vggface2: A dataset for recognising faces across pose and age*, 2018. arXiv: 1710.08092 [cs.CV]. [Online]. Available: <https://arxiv.org/abs/1710.08092>.
- [198] T. Unterthiner, S. van Steenkiste, K. Kurach, R. Marinier, M. Michalski, and S. Gelly, *Towards accurate generative models of video: A new metric & challenges*, 2019. arXiv: 1812.01717 [cs.CV]. [Online]. Available: <https://arxiv.org/abs/1812.01717>.
- [199] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, “Image quality assessment: From error visibility to structural similarity,” *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004. DOI: 10.1109/TIP.2003.819861.
- [200] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, “Robust speech recognition via large-scale weak supervision,” 2022. [Online]. Available: <https://arxiv.org/abs/2212.04356>.
- [201] W. contributors, *Whisperspeech: An open-source text-to-speech system built by inverting whisper*, <https://github.com/WhisperSpeech/WhisperSpeech>, Accessed: 2025-12-03, 2024.

- [202] T. Feng, D. Dimitriadis, and S. Narayanan, “Can synthetic audio from generative foundation models assist audio recognition and speech modeling?” *arXiv preprint arXiv:2406.08800*, 2024.
- [203] S. Minaee *et al.*, *Large language models: A survey*, 2025. arXiv: 2402.06196 [cs.CL]. [Online]. Available: <https://arxiv.org/abs/2402.06196>.
- [204] U. Cappellazzo *et al.*, “Large language models are strong audio-visual speech recognition learners,” in *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2025, pp. 1–5.
- [205] S. Yin *et al.*, “A survey on multimodal large language models,” *National Science Review*, vol. 11, no. 12, Nov. 2024, ISSN: 2053-714X. DOI: 10.1093/nsr/nwae403. [Online]. Available: <http://dx.doi.org/10.1093/nsr/nwae403>.
- [206] A. Haliassos, R. Mira, H. Chen, Z. Landgraf, S. Petridis, and M. Pantic, “Unified speech recognition: A single model for auditory, visual, and audiovisual inputs,” *Advances in Neural Information Processing Systems*, vol. 37, pp. 139 673–139 699, 2024.
- [207] B. F. Labs *et al.*, *Flux.1 kontext: Flow matching for in-context image generation and editing in latent space*, 2025. arXiv: 2506.15742 [cs.GR]. [Online]. Available: <https://arxiv.org/abs/2506.15742>.
- [208] W. Kong *et al.*, “Hunyuanvideo: A systematic framework for large video generative models,” *arXiv preprint arXiv:2412.03603*, 2024.
- [209] Z. Chen *et al.*, *Dip: Taming diffusion models in pixel space*, 2025. arXiv: 2511.18822 [cs.CV]. [Online]. Available: <https://arxiv.org/abs/2511.18822>.
- [210] Q. Wang, M. Wang, F. Jiang, Y. Fan, Y. Qi, and M. Xu, *Fantasyportrait: Enhancing multi-character portrait animation with expression-augmented diffusion transformers*, 2025. arXiv: 2507.12956 [cs.CV]. [Online]. Available: <https://arxiv.org/abs/2507.12956>.