Imperial College London

DEVELOPMENT OF AN INTELLIGENT FLEXIBLE SHEET METAL FORMING TECHNIQUE

By

Shiming LIU

A Thesis submitted to the
Department of Mechanical Engineering
Imperial College London

In conformity with the requirements for the Degree of Doctor of Philosophy and for the Diploma of the Imperial College London

October 2022

Abstract

This thesis presents an intellectualisation solution for an automated flexible sheet metal forming process, which produces arbitrarily curved sheet metal components through incremental local deformation. For the first time, an intelligent flexible sheet metal forming system is established to provide an end-to-end tool path planning framework, with state-of-the-art theory-guided/data-driven artificial intelligence (machine learning) technologies embedded for tool path fast prediction and system self-improvement. The system is composed of seven parallel modules covering data storage, raw data pre-processing, tool path planning, process computation and validation, post-processing and feedback mechanism, of which the tool path planning accuracy and efficiency is ensured through a dual modular redundancy-akin approach. The path planning problem is investigated through three stages: target sheet deformed by 1) a single punch, 2) multiple punches, and 3) unlimited punches. At the first stage, a classic convolutional neural network (CNN) is developed for process parameter prediction of bending processes. A novel theory-guided regularisation method, utilising Swift's law as guidance, for deep neural network (DNN) training is proposed, which has been verified to precede conventional data-driven method in severely scarce data condition. The theory-guided DNN exhibits a more robust generalisation capability and learning consistency than the data-driven DNN in experiments with different training data structures, workpiece materials and sheet metal forming applications. At the second stage, a consecutive rubber-tool forming process is developed to incrementally deform the sheet metal, whose tool path is predicted through a recursive tool path prediction framework. The framework adopts a deep learning model to learn the topological relationship between the current and target workpiece shape. Three series of state-of-the-art deep learning models, namely single feature extractor, cascaded networks and long short-term memory (LSTM) models are investigated. With over 6000 samples, the CNN LSTMs have been verified to be the most superior in tool path planning owing to their ability in learning both spatial and temporal features, while the other two models exhibit inferior performances when path planning complexity increases. At the third stage, deep reinforcement learning (DRL) is exploited to learn the optimal tool path, without expertise in prior, for arbitrarily curved sheet metal. In this context, a generalisable tool path planning strategy is proposed to address the commonly-known low generalisation issue of pure DRL approach. The

strategy factorises workpiece into segments and optimises tool path through dynamic programming, exploiting both DRL and deep supervised learning (DSL) technologies. The strategy manifests self-learning characteristics during training, and it has been verified applicable to tool path planning of arbitrary sheet metal parts through a case study. With the investigation results and dedicated tool path planning strategy in this thesis, the intelligent flexible sheet metal forming system can be deployed for fast and accurate tool path generation for prototyping during the sheet metal production cycle.

Statement of Originality

I hereby declare that this thesis 'Development of An Intelligent Flexible Sheet Metal Forming Technique' and the work reported herein was composed by and originated entirely from me. Information derived from the published and unpublished work of others has been acknowledged in the text and references.

Shiming LIU

Copyright Declaration

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

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

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

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

Shiming LIU

Acknowledgements

First and foremost, I would like to express my heartfelt gratitude to my supervisors, Dr. Zhusheng Shi and Prof. Jianguo Lin, who have relentlessly given me constructive guidance and enthusiastic encouragement throughout my whole PhD study. I would like to thank Prof. Lin for kindly offering me this great opportunity to pursue my PhD degree in such a harmonious world-leading research group, only I know what this opportunity means to me. His nuanced understanding of metal forming and penetrating insights into its future development as well as passionate and vigorous attitude towards research have deeply inspired and invigorated me. I would like to express my deep gratitude to Dr. Shi, to whom I owe an important debt for his meticulous care and tremendous help in every possible way. His generous and exhaustive impartation of priceless academic and industrial experience has definitely consolidated my research work. My sincere appreciation also goes to his rigorous supervision on each research milestone and attentive proofreading for each research paper.

I would like to gratefully thank Prof. Hui Yu and Dr. Yifan Xia, from University of Portsmouth (and Dr. Xia has recently joined Shandong University), for their tight collaborations and precious advice on this interdisciplinary research from the perspective of artificial intelligence. Those enlightening meetings and illuminating discussions have contributed to my PhD research to a great extent. I wish to thank my colleague, Mr. Haosu Zhou, for those passionate discussions and brainstorm we had about the future of artificial intelligence in metal forming. My acknowledgement also goes to the staffs, Ms. Kate Lewis, Ms. Val Crawford and Mr. Alex Toth, for their administrative and experimental support during my PhD study.

I would like to thank China Scholarship Council (CSC) for their subsistence allowance throughout my PhD study.

My appreciation also goes to my colleagues and friends, Dr. Xiaochuan Liu, Dr. Qunli Zhang, Dr. Jun Liu, Dr. Wenbin Zhou, Ms Liwei Yang, Mr. Xiaochen Lu, Dr. Yang Zheng, Dr. Yuehan Liu, Dr. Jinghua Zheng, Dr. Qinmeng Luan, Dr. Qi Rong, Mr. Chenpeng Tong, Mr. Xi Wang, Ms. Jiaxin Lv, Mr. Shuai Jiang and Mr. Xiao Hu. The memories of us endeavouring, laughing, bragging and travelling together are lifelong treasures to me. Particularly, I wish to

appreciate Dr. and Mrs. Zhou for their generous and frequent dinner invitations during the pandemic.

Lastly but most importantly, I would like to express my deepest gratitude to my parents. It is your firm support and unquantifiable love that encourage me to start my PhD journey and enjoy the views along the way. I love both of you a thousand times and wish you all the best.

Shiming LIU

Table of Contents

Abstract 2
Statement of Originality 4
Copyright Declaration 5
Acknowledgements 6
Table of Contents 8
List of Figures 12
List of Tables 20
Nomenclature 23
Abbreviation 26
Publications related to this thesis 29
Chapter 1 Introduction 30
1.1 Advanced sheet metal stamping technologies and applications 30
1.2 An alternative approach: free-form flexible sheet metal stamping 32
1.3 Aims and approach 35
1.4 Thesis structure 36
Chapter 2 Literature Review 38
2.1 Conventional sheet metal stamping 38
2.1.1 Cold, warm and hot stamping 38
2.1.2 Springback 42
2.2 Flexible sheet metal free forming – automated hammering 44
2.3 Machine learning: supervised learning and applications 47
2.3.1 Data-driven learning 48
2.3.2 Theory-guided learning 50

2.3.3 Deep neural network 50
2.3.4 Applications 54
2.4 Machine learning: reinforcement learning and applications 56
2.4.1 Q-learning algorithms 57
2.4.2 Policy gradient algorithms 58
2.4.3 Applications 60
2.5 Machine learning applications to sheet metal stamping process 61
2.5.1 Process monitoring and fault diagnosis 62
2.5.2 Surrogate assisted optimisation 65
2.5.3 Tool path planning 68
2.6 Summary 71
Chapter 3 Intelligent flexible sheet metal forming system architecture 74
3.1 Introduction 74
3.2 The overall architecture 76
3.3 Deep learning applications to main modules 78
3.4 Summary 80
Chapter 4 Single punch: theory-guided deep learning with scarce experimental data 82
4.1 Introduction 82
4.2 Experimental setup and forming digitisation 83
4.2.1 Experimental setup 83
4.2.2 Forming digitisation 84
4.3 Theory-guided deep learning and neural networks 86
4.4 Data acquisition and training method 90
4.4.1 Data acquisition 90
4.4.2 Training method 91
4.5 Learning results 92

4.5.1 Randomly sampled data 92
4.5.2 Scarce data 93
4.5.3 Severely scarce data 95
4.6 Summary 100
Chapter 5 Multiple punches: tool path planning with deep supervised learning technology 102
5.1 Introduction 102
5.2 Forming experiments test setup 103
5.3 Process digitisation and data acquisition 105
5.4 Consecutive tool path prediction framework and its training method 108
5.5 Deep learning models 110
5.5.1 Single feature extractor 111
5.5.2 Cascaded networks 111
5.5.3 LSTM models 115
5.6 Learning results 117
5.6.1 Dataset within 2 punches 118
5.6.2 Dataset within 3 punches 123
5.7 LSTM performance analysis 126
5.8 Summary 127
Chapter 6 Unlimited punches: a generalisable tool path planning strategy 129
6.1 Introduction 129
6.2 A preliminary study on reinforcement learning application to one single deformation step 130
6.2.1 FE modelling 130
6.2.2 Deep reinforcement learning setup 131
6.2.3 Learning results 133

6.2.4 Discussion 136
6.3 Generalisable tool path planning strategy 137
6.4 Summary 139
Chapter 7 Instantiation of the generalisable tool path planning strategy 140
7.1 Introduction 140
7.2 Experiments test setup of consecutive forming process and forming digitisation 140
7.3 Forming goal and forming parameters design 142
7.4 Deep reinforcement learning setup and virtual environment 144
7.4.1 Learning setup and hyperparameters 144
7.4.2 Virtual environment for RL algorithms comparison 147
7.5 Deep supervised learning models and training methods 147
7.5.1 Deep neural networks 148
7.5.2 Training method and hyperparameters 148
7.6 Learning results 149
7.6.1 Selection of reinforcement learning algorithm 149
7.6.2 Tool path learning for 25 workpieces using Double-DQN 154
7.6.3 Tool path generalisation using supervised learning model 159
7.7 Case study verifying the generalisable tool path planning strategy 164
7.8 Summary 166
Chapter 8 Conclusions and future work 168
8.1 Conclusions 168
8.2 Future work 171
References 172
Appendix A. Virtual environment configuration 184
Appendix B. Arbitrary generation of workpiece segments 185
Appendix C. Extraordinarily large CPE2 186

List of Figures

Figure 1-1. Applications of sheet metal components in automobile, aircrafts and high-speed trains. 30

Figure 1-2. The V-shaped stamping tools in a cold stamping process [7]. 31

Figure 1-3. Two basic hot stamping processes: direct and indirect hot stamping [12]. 32

Figure 1-4. Four commonly used tools by the metal smith on the shop floor [18]. 33

Figure 1-5. Manual and automated sheet metal hammering processes. 34

Figure 1-6. Flowchart of the thesis structure. 36

Figure 2-1. Forming temperatures for different forming processes [4]. 39

Figure 2-2. Stress-strain curves of aluminium alloys with different tempers [4]. 39

Figure 2-3. Experimental setup of warm stamping [28]. 40

Figure 2-4. Effect of forming temperature on stamping performance. 41

Figure 2-5. Sheet metal temperature profile during HFQ® process [4]. 41

Figure 2-6. Springback angle notations for different bending processes. 42

Figure 2-7. Springback in practice for different bending processes. 43

Figure 2-8. The automated hammering system for sheet metals [41]. 45

Figure 2-9. A single-point incremental sheet forming process [44]. 45

Figure 2-10. An automated ISP process with its (a) hammering equipment, (b) forming setup and (c) forming result [43]. 46

Figure 2-11. New automated hammering process with (a) centre punch and adjustable dies and (b) fixtureless manipulation. 47

Figure 2-12. Three-layer fully connected neural network (adapted from [65]). 51

Figure 2-13. Architecture of a convolutional neural network [69]. 51

Figure 2-14. Architecture of RNN: rolled form (left) and unrolled form (right). 53

Figure 2-15. Architecture of LSTM. 53

Figure 2-16. A deep learning application for image captioning, integrating work from both CV and NLP [83]. 54

Figure 2-17. Reinforcement learning process. 56

Figure 2-18. The t-SNE visualisation of the representations extracted from the last hidden layer of the learning agent, coloured by the state value [96] (black colour indicates the highest state value). 61

Figure 2-19. Neural networks for process monitoring of a deep drawing process [115]. 62

Figure 2-20. Different bases for tool wear monitoring in a stamping process. 63

Figure 2-21. The formed part through S-Rail forming process and the surrogate model for its springback angles prediction [132]. 66

Figure 2-22. Res-SE-U-Net: an image-based surrogate model for sheet metal stamping [137]. 67

Figure 2-23. The neural network and data processing in an automated driving process. 69

Figure 2-24. The experimental test setup and data processing in an automated wheeling process [144]. 69

Figure 2-25. The sheet metal deformation progress through the ISF process [145]. 70

Figure 2-26. Tool path with predicted (a) support force and (b) support angle for an ISF process [146]. 71

Figure 3-1. The concept of realising an intelligent sheet metal forming system through machine learning technologies. 75

Figure 3-2. The overall architecture of the intelligent sheet metal forming system. 76

Figure 3-3. The architecture of the Process Generation Module. 78

Figure 3-4. The architecture of the Process Learning Module. 79

Figure 3-5. The architecture of the Process FE Module. 80

Figure 4-1. The geometry, test setup and plane strain FE model for the four-point bending and air bending. 83

Figure 4-2. Examples of target components to be manufactured using four-point bending and air bending respectively and geometry representation for assessing the error between the designed and manufactured components. 85

Figure 4-3. Examples of workpiece geometry representations (801 pixels). 85

Figure 4-4. The relative error between the calculated results of the theory-guided penalty term for the whole 201 examples and the ideal value (zero) for the indicated materials and processes. 4-pt bend denotes four-point bending process. 88

Figure 4-5. Theory-guided learning data flow. 89

Figure 4-6. Architecture of the deep neural network. 89

Figure 4-7. Training data acquisition and data flow between FE simulation and TG-DNN training. 90

Figure 4-8. Comparison between the true data and predictions of the data domain from SVR, ST-DNN, MT-DNN and TG-DNN, which were trained with different amount of data. The true data were obtained from four-point bending test of AA6082. The training data were randomly sampled from the whole dataset. 92

Figure 4-9. The training loss curves for the three DNNs (upper row) and the evaluations of the DNNs on the total data domain (lower row) after training with about 20% training data acquired from different applications. The training data was designed to be symmetric and asymmetric to have large continuous regions of missing data domain. 95

Figure 4-10. Comparison of generalisation capability on test set between the data-driven and theory-guided DNNs with a) full data b) half data, obtained from four-point bending process of AA6082 workpiece. 96

Figure 4-11. The training loss curves for the data-driven and theory-guided DNNs (upper row) and the evaluations of all the DNNs on the test set (lower row) after training with 1/4 training data acquired from different applications. 97

Figure 4-12. Comparison between the shape of the target workpiece (with target stroke of 24 mm) and the workpiece manufactured with the stroke predicted by the data-driven and theory-guided DNNs, which were trained with 1/4 symmetric data acquired from mid-point of the AA6082 workpiece from (a) four-point bending and (b) air bending. 99

Figure 4-13. Comparison between the training and test loss of physical properties ($\boldsymbol{\sigma v}$ and $\boldsymbol{\varepsilon eq}$) between well-trained MT-DNN and TG-DNN with 1/4 symmetric training data acquired from mid-point of the AA6082 workpiece from 4-point bending. 100

Figure 5-1. The test setup, plane strain FE model and FE mesh in forming test. 103

Figure 5-2. Punch locations and rubber-tool forming process. 105

Figure 5-3. Examples of workpiece representations in graph drawn by coordinates data (left) and binary image (right). 106

Figure 5-4. Illustration of possible punch locations and stroke options. 107

Figure 5-5. Infeasible workpiece shape with insufficient opening for punch in (a) and self-intersection in (b). 107

Figure 5-6. Data pipeline through deep learning models at the training phase (solid square) and application phase (dashed square). The filled arrows denote the data flow during the training of the deep learning model. 109

Figure 5-7. Data pipeline through the cascaded network. A target workpiece “301-9mm_501-5mm_701-9mm” is used as an example for illustration. The filled arrows denote the data flow during the training of the image reconstruction model. The top long unfilled arrow denotes that the “current” workpiece is also an input to the CNN model. The CNN model is pre-trained in prior and implemented as a feature extractor in the cascaded network. 111

Figure 5-8. Network architectures and details of parameters of the six image reconstruction models used in the cascaded network. Conv and pooling layers are defined in the format of (channels, kernel, stride) and (pool, stride), respectively. ConvT denotes transposed convolution layer. In all architectures, all convolutional layers are followed by a ReLU activation layer, which is not denoted in figures for brevity. “Same padding” is used in conv and convT for all models. As the learning is configured as a classification problem, the outputs of the models have two channels. 114

Figure 5-9. Data pipeline through the CNN LSTM. A target workpiece “301-9mm_501-5mm_701-9mm” is used as an example for illustration. The input includes the workpiece shapes at the first three deformation time steps and the model predicts the forming parameters at the fourth time step. 116

Figure 5-10. The architecture of single LSTM model (without layers in dashed square) and stacked LSTM model (with layers in dashed square). The four inputs at four time steps to the LSTM has dimension of 288×576×1. The LSTM layer has 256 units 116

Figure 5-11. t-SNE feature representation for Dataset within 3 punches. 117

Figure 5-12. The IoU results distribution over its whole score domain from six image reconstruction models. 120

Figure 5-13. Bar chart of prediction accuracy from all models on training set and test set of the Dataset within 2 punches, respectively. The learning results are averaged from all three trainings and the error bar shows the performance variation. All trainings are provided with the same amount of raw data in the original workpiece image dataset, and these data are reallocated and assembled to conform to different model architectures. 122

Figure 5-14. Bar chart of prediction accuracy from all models on training set and test set of the Dataset within 3 punches, respectively. The learning results are averaged from all three trainings and the error bar shows the performance variation. All trainings are provided with the same amount of raw data in the original workpiece image dataset, and these data are reallocated and assembled to conform to different model architectures. 125

Figure 5-15. The training and test accuracy on each punch location and stroke from single and stacked LSTM models. The accuracies were averaged from all trainings. 126

Figure 6-1. The 3-D model and FE model of the hammering process. 130

Figure 6-2. DQN architecture for the RL process. 132

Figure 6-3. Overall learning architecture embedded with deep Q-learning. 132

Figure 6-4. RL training process in terms of batch loss and total reward per episode. 134

Figure 6-5. Q-values versus training steps for hammer at (a) initial position (x, y) = (0, 0); (b) (x, y) = (0, -5). 135

Figure 6-6. Q-values versus training steps for hammer at (a) (x, y) = (-5, 0); (b) (x, y) = (-5, -5)..... 135

Figure 6-7. Q-values versus training steps for hammer at (x, y) = (-10, -5). 136

Figure 6-8. Schematic digitisation procedure for workpiece state representation $\Delta \boldsymbol{K}$-graph and the classification of three groups of segments. The drawings for target and current workpieces

depict their top surfaces. The dashed lines in Group 2 signify other segments having the same shape features as the solid line, which are also counted in this group. L-length denotes longitudinal length..... 137

Figure 6-9. The generalisable tool path planning strategy through deep reinforcement and supervised learning..... 138

Figure 7-1. Test setup and FE model for the rubber-tool forming. The lengths of the deformation and trim zones are 30 mm and 10 mm, respectively..... 141

Figure 7-2. Example of workpiece shape (left) and its curvature K distribution along node locations (right). The region highlighted in red in the drawing denotes the deformation zone, and the K-graph is generated from the workpiece top surface of this zone..... 142

Figure 7-3. Close punch effect 1 on the punch with stroke of 3.0 mm at the location of 151. (a), (b) and (c) present the $\Delta\boldsymbol{K}$-graphs before and after this punch on the workpiece which has been consecutively deformed, in prior, by 2, 3 and 4 punches, respectively. 143

Figure 7-4. Close punch effect 2 from the punches near location of 118. (a) and (b) present the $\Delta\boldsymbol{K}$-graphs before and after 1 punch and 50 punches, respectively. The area highlighted by dashed circle is where CPE2 was found..... 143

Figure 7-5. Reward function and its evaluation for each action. (a) shows the reward evaluation method for punch stroke of 2.2 mm at location of 162, and the two lines denote the initial and current $\Delta\boldsymbol{K}$-graph; (b) shows the reward function for evaluation. 144

Figure 7-6. The reinforcement learning process for the tool path learning of the consecutive rubber-tool forming process, of which the FE simulation (FE sim) was used as the RL environment to provide real-time deformation results. The vertical line in the $\Delta\boldsymbol{K}$-graph denotes the punch location. 145

Figure 7-7. Comparison of the performances of three Q-learning algorithms trained at 10^{-2} learning rate, with and without heuristic, for tool path learning in terms of termination step. The upper and lower dashed lines denote the average termination step, estimated from Gaussian process regression, through the training process of the algorithms implemented with and without Heuristic 2, respectively. The shaded regions denote 95% confidence interval. The unit for training steps, K, denotes 10^3. 150

Figure 7-8. Comparison of the performances of 6 reinforcement learning algorithms studied in this research in terms of termination step. The solid and dashed lines denote the average termination step, estimated from Gaussian process regression, through the training process of the algorithms. The shaded regions denote 95% confidence interval. 153

Figure 7-9. Comparison between RL using virtual environment (VE) and FE simulation environment (FE) in terms of (a) the history of steps in each episode, (b) the history of total rewards in each episode and (c) the tool path predictions..... 154

Figure 7-10. The (a) K-graphs and (b) workpiece shapes for all the target workpieces generated. 154

Figure 7-11. The history of the maximum $\Delta \boldsymbol{K}$ value at the end of each episode (Ep) throughout the learning process. The arrow shows the learning progress of effective tool path. 155

Figure 7-12. Examples showing self-learning characteristics of (a) improving tool path efficiency and (b) circumvention of overpunch. The results were from step 31 at three different episodes of the tool path learning process for a workpiece..... 156

Figure 7-13. The two-dimensional embedding, generated through t-SNE, of the representations in the last hidden layer of the Double-DQN to workpiece states ($\Delta \boldsymbol{K}$-graphs) experienced during tool path learning. The points are coloured according to stroke values selected by the agent. The graph at the top left corner shows the initial $\Delta \boldsymbol{K}$-graph, and the axis labels of the other $\Delta \boldsymbol{K}$-graphs (numbered from ① to ⑪) are omitted for brevity. The vertical lines in the $\Delta \boldsymbol{K}$-graphs denote the punch locations. 157

Figure 7-14. An example of (a) tool path learned by RL and (b) its top view and side view. The height of the bars in (a) were proportionally decreased for better visualisation. 158

Figure 7-15. An example of workpiece deformed by the tool path predicted by the Double-DQN. Top: the dimension error between the target workpiece and the final workpiece deformed along the predicted tool path. Bottom: the workpiece shape at each forming step (dotted line) compared to its target shape (solid line). Forming step of 0 denotes blank sheet..... 159

Figure 7-16. The length of the tool path learned through Double-DQN for each variant of workpiece segment. 159

Figure 7-17. The generalisation loss curves along the training processes of the LSTM models with VGG16, ResNet34 and ResNet50 as the feature extractor, respectively. The number 25 and 20 in the parenthesis denote the amount of tool paths used for training. 160

Figure 7-18. The learning performance of LSTMs trained with 25 tool paths data (solid squares) and 20 tool paths data (dashed squares) on a test workpiece. For the former, the prediction results from LSTMs with (a) VGG16, (b) ResNet34 and (c) ResNet50 are presented, while for the latter, that from (d) VGG16 LSTM are presented. The prediction results include three parts as followed. Top: the tool path prediction from LSTMs (SL) and its comparison to the tool path from reinforcement learning (RL); Mid: the final $\Delta \boldsymbol{K}$-graph after deformation; Bottom: the dimension error and the comparison between the deformed workpiece shape and its target. 162

Figure 7-19. Evaluation of the pure RL strategy by assessing the tool path prediction results from the trained Double-DQN model for new variants of workpiece. The $\Delta \boldsymbol{K}$-graphs were acquired after the first punch, predicted by the RL model, for the variant used for tool path learning through RL (solid line) and new variants that were never seen in the learning process (dashed line). 164

Figure 7-20. The entire tool path for a new workpiece, which is aggregated by the subpaths predicted by the VGG16 LSTM for each segment of the workpiece. The three segments were never seen in the LSTM training. 165

Figure 7-21. The comparison between the workpiece shape deformed through the generalisable tool path planning strategy and its target shape. (a) deformed workpiece shape yielded by the strategy and (b) workpiece shape after two supplementary punches near the junctions of the three workpiece segments. 166

List of Tables

Table 4-1. Experimental setup data for four-point bending and air bending (unit: mm). 84

Table 4-2. Values of the parameters for the materials investigated. 84

Table 4-3. Hyperparameters search space 92

Table 4-4. Comparison of the average of the maximum distance between the target and predicted workpiece shape ($\Delta GEOM_{pred}$) from the well-trained SVR and DNNs evaluated on the total data domain (unit: mm). 93

Table 4-5. Comparison of the average of the maximum distance between the target and predicted workpiece shape ($\Delta GEOM_{pred}$) from the well-trained DNNs evaluated on the total data domain in all experiments. 95

Table 4-6. Comparison of the average of the maximum distance between the target and predicted workpiece shape ($\Delta GEOM_{pred}$) from the well-trained data-driven and theory-guided DNNs evaluated on the test set. The data were obtained from four-point bending process of AA6082 workpiece (unit: mm). 97

Table 4-7. Comparison of the average of the maximum distance between the target and predicted workpiece shape ($\Delta GEOM_{pred}$) from the well-trained data-driven DNNs and theory-guided DNN evaluated on the test set in all experiments. 98

Table 5-1. Specification of the test setup of rubber-tool forming. 104

Table 5-2. Material properties. 104

Table 5-3. Key parameters for CNN training with Dataset within 2 punches. 118

Table 5-4. Learning results from both CNN models trained with Dataset within 2 punches on the prediction of tool path. 118

Table 5-5. Key parameters for training with the new dataset without forming sequence. 119

Table 5-6. Learning results from both CNN models trained with new dataset without forming sequence. 119

Table 5-7. Key parameters for training image reconstruction models with Dataset within 2 punches. 120

Table 5-8. The prediction accuracy on forming parameters from the cascaded models embedded with six different image reconstruction models with Dataset within 2 punches. 121

Table 5-9. Key parameters for training LSTM models with Dataset within 2 punches. 121

Table 5-10. Learning results from single and stacked LSTM models trained with Dataset within 2 punches on the prediction of tool path. 122

Table 5-11. key parameters for training the single feature extractors with Dataset within 3 punches. 123

Table 5-12. Learning results from single feature extractors trained with Dataset within 3 punches on the prediction of tool path. 123

Table 5-13. Key parameters for training the cascaded models with Dataset within 3 punches. 124

Table 5-14. Learning results from cascaded models trained with Dataset within 3 punches on the prediction of tool path. 124

Table 5-15. key parameters for training the LSTM models with Dataset within 3 punches. 124

Table 5-16. Learning results from LSTM models trained with Dataset within 3 punches on the prediction of tool path. 125

Table 6-1. Test setup details for hammering modelling (unit: mm). 130

Table 6-2. FE modelling details for the hammering process. 131

Table 6-3. Prescribed actions for the hammer (unit for x/y: mm; unit for Vel: mm/s)..... 131

Table 6-4. Grid-independence study results. 133

Table 7-1. Test setup details for the rubber-tool forming. 141

Table 7-2. Learning methods for the six reinforcement learning algorithms (Y – True, N – False). 146

Table 7-3. The learning hyperparameters for training the six RL algorithms. step/ep denotes step per episode and freq. denotes frequency. 146

Table 7-4. Key parameters for DNN training. 149

Table 7-5. Performance comparison of the Q-learning algorithms for tool path learning. Term. denotes termination. Cvg. denotes converge. Freq. denotes frequency. Avg. denotes average. "-" denotes less than 3 terminations occur in the training..... 151

Table 7-6. Performance comparison of the policy gradient algorithms for tool path learning. Term. denotes termination. Cvg. denotes converge. Freq. denotes frequency. Avg. denotes average. "-" denotes less than 3 terminations occur in the training. 151

Nomenclature

Variables	*Unit*	*Specification*
a	-	Action of agent in reinforcement learning
$\boldsymbol{\mathcal{A}}$	-	A set of actions
A	-	Advantage-value in reinforcement learning
$\hat{A}$	-	Generalised Advantage Estimate in reinforcement learning
$\boldsymbol{b}$	-	Bias vector in neural network
$\boldsymbol{c}$	-	Cell state vector in LSTM
E	GPa	Young's modulus of material
$f(\cdot)$	-	Activation function in neural network
f_t	-	Forget gate in LSTM
$g(\cdot)$	-	Convolution function in neural network
G	%	Level of forming goal achievement in tool path planning
$\bar{G}$	%	Average level of forming goal achievement in tool path planning
$\boldsymbol{h}$	-	Hidden state vector in LSTM
H	-	Height of pooling in CNN
i_t	-	Input gate in LSTM
$I(\cdot)$	-	Input function in neural network
$J(\cdot)$	-	Objective function for optimisation/neural network training
$\hat{J}(\cdot)$	-	Regularised objective function for optimisation/neural network training
K	-	Curvature of sheet metal surface
$K(\cdot)$	-	Kernel function in CNN
$\boldsymbol{K}$	-	Curvature distribution of sheet metal surface
$\Delta\boldsymbol{K}$	-	Curvature difference distribution of sheet metal surface
K_s	-	Strength coefficient in material model
L	-	Loss for optimisation/neural network training
n	-	Strain-hardening coefficient in material model

o_t	-	Output gate in LSTM
p	-	Probability
$\boldsymbol{P}$	-	Transition probability model in reinforcement learning
Q	-	Action-value in reinforcement learning
r	-	Reward in reinforcement learning
$\boldsymbol{R}$	-	A set of rewards
s	-	State in reinforcement learning
$\boldsymbol{S}$	-	A set of states
V	-	State-value in reinforcement learning
$\boldsymbol{w}$	-	Weights vector in neural network
$\boldsymbol{W}, \boldsymbol{U}$	-	Weights matrix in neural network
$\boldsymbol{x}$	-	Input vector to neural network
$\boldsymbol{X}$	-	Input matrix to neural network
$\boldsymbol{y}$	-	Target vector for neural network
$\boldsymbol{Y}$	-	Target matrix for neural network
$\widehat{\boldsymbol{y}}$	-	Model prediction vector from neural network
$\widehat{\boldsymbol{Y}}$	-	Model prediction matrix from neural network
$\boldsymbol{\theta}$	-	Model trainable parameters in neural network
$\boldsymbol{\Theta}$	-	Model trainable parameters and hyperparameters in neural network and its training
Ω	-	Norm penalty term in objective function
η	-	Learning rate for neural network training
γ	-	Discounting ratio for neural network training
γ_p	-	Poisson's ratio of material
$\pi(\theta)$	-	Stochastic policy model in reinforcement learning
$\mu(\theta)$	-	Deterministic policy model in reinforcement learning
μ	-	Friction coefficient between surfaces
ϵ	-	Epsilon in ϵ-greedy policy in reinforcement learning
ϵ'	-	Clipping ratio in PPO algorithm
σ	MPa	True stress

σ_v	MPa	von Mises stress
ε	-	True strain
ε_{eq}	-	Equivalent plastic strain
ρ	g/cm^3	Density of material
$\Delta GEOM_{pred}$	mm	Maximum distance between the target and predicted workpiece shape

Abbreviation

A2C	Advantage actor-critic
ANN	Artificial neural network
CAE	Convolutional autoencoder
CFD	Computational fluid dynamics
CNN	Convolutional neural network
CPE	Close punch effect
CV	Computer vision
DB	Database
DDPG	Deep deterministic policy gradient
DNN	Deep neural network
Double-DQN	Double deep Q-network
DQN	Deep Q-network
DRL	Deep Reinforcement learning
DSL	Deep supervised learning
Dueling-DQN	Dueling deep Q-network
Ep	Episode
FCN	Fully convolutional network
FEA	Finite element analysis
FEB	Forming experience base
GA	Genetic algorithm
GAE	Generalised advantage estimate

GAN	Generative adversarial network
GDM	Geometry digitisation module
GNN	Graphical neural network
GRU	Gated recurrent unit
IBM	Image-based method
IoU	Intersection over union
ISF	Incremental sheet forming
ISP	Incremental sheet punching
KB	Knowledge base
KBS	Knowledge-based system
KNN	K-nearest neighbours
LSTM	Long short-term memory
MAE	Mean absolute error
MC	Monte Carlo
MCTS	Monte Carlo tree search
MDP	Markov decision process
MLP	Multilayer perceptron
MSE	Mean square error
MT-DNN	Multi-task DNN
NLL	Negative log-likelihood
NLP	Natural language processing
PDE	Partial differential equations
PFEM	Process FE module
PGM	Process generation module

PINN	Physics-informed neural network
PLM	Process learning module
PPO	Proximal policy optimisation
ReLU	Rectified linear unit
RNN	Recurrent neural network
SBM	Scalar-based method
SGD	Stochastic gradient descent
ST-DNN	Single-task DNN
SVM	Support vector machines
SVR	Support vector regressor
TG-DNN	Theory-guided DNN
THLD	Threshold
VE	Virtual environment

Publications related to this thesis

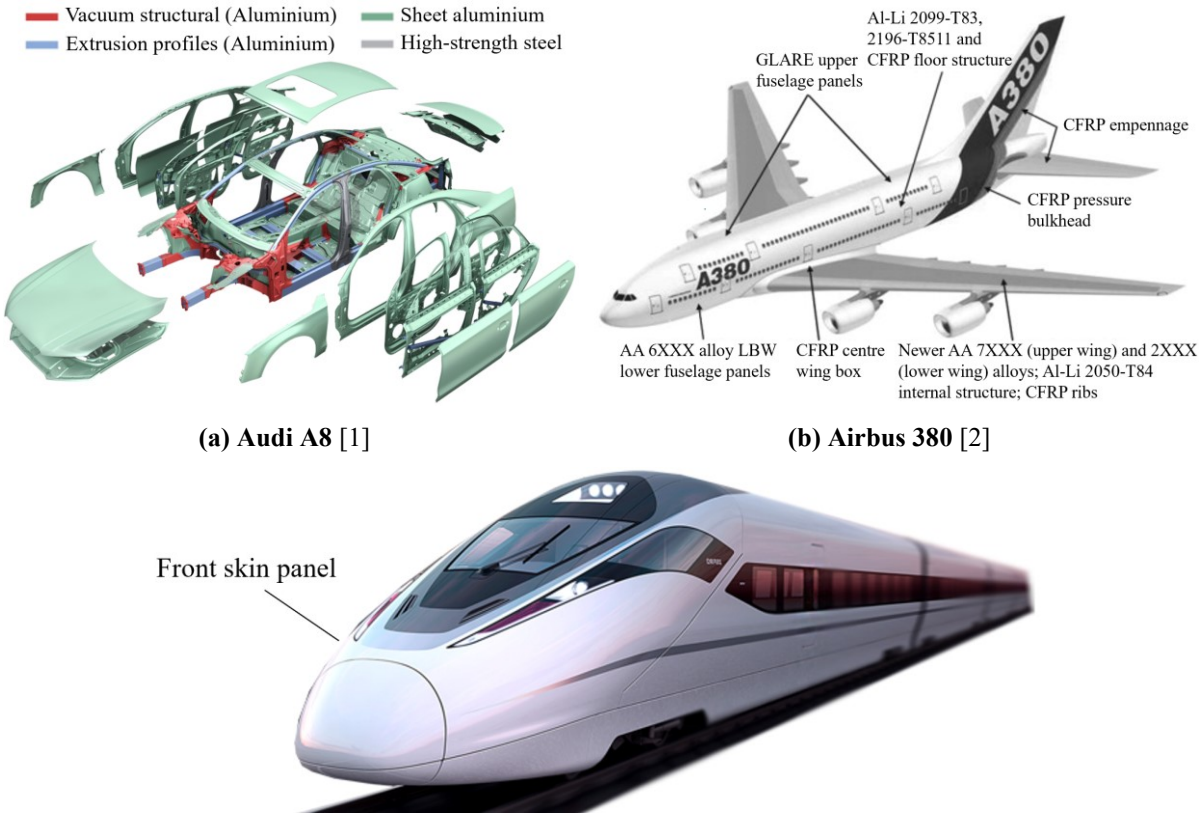
1. **S. Liu**, Z. Shi, J. Lin, H. Yu, A generalisable tool path planning strategy for consecutive free-form sheet metal stamping through deep reinforcement and supervised learning, *Under review*.
2. **S. Liu**, Y. Xia, Y. Liu, Z. Shi, H. Yu, Z. Li, J. Lin, Tool path planning of consecutive free-form sheet metal stamping with deep learning, J. Mater. Process. Technol. 303 (2022) 117530.
3. **S. Liu**, Y. Xia, Z. Shi, H. Yu, Z. Li, J. Lin, Deep learning in sheet metal bending with a novel theory-guided deep neural network, IEEE/CAA J. Autom. Sin. 8 (2021) 565–581.
4. **S. Liu**, Z. Shi, J. Lin, Z. Li, Reinforcement learning in free-form stamping of sheet-metals, Procedia Manuf. 50 (2020) 444–449.

Chapter 1

Introduction

1.1 Advanced sheet metal stamping technologies and applications

Sheet metal components are nowadays ubiquitous in various industrial products, such as automobile, aircrafts and high-speed trains as shown in Figure 1-1. Benefit from the short forming cycle of contemporary advanced sheet metal stamping techniques, which make it feasible for the mass production of lightweight sheet metals, the manufacturing budgets are constantly reduced, and a burgeoning era of industrialisation arises.



(a) Audi A8 [1]

(b) Airbus 380 [2]

(c) High-speed train [3]

Figure 1-1. Applications of sheet metal components in automobile, aircrafts and high-speed trains.

Sheet metal stamping techniques can be classified as cold stamping and hot stamping. As indicated by their names, the two techniques differ in the working temperature during forming. In the manufacturing industry of aluminium alloy components, cold stamping is believed to be the most widely used technique [4]. During the cold stamping process, the sheet blank is first clamped by blank holder. It is then stamped by the punch and die at room temperature, followed by the relaxation of load for springback [5]. Figure 1-2 shows several punches and dies in a cold stamping process and their test setup in experiments. This technique is favourable for its short production cycle and environmentally friendly forming process [6]. However, due to the stress relaxation after forming, notably large springback inevitably occurs for high strength aluminium alloys and steels after cold stamping [7,8]. In addition, the additional heat treatment after cold stamping for restoring the microstructure and mechanical properties can incur extra springback [9] and distortions [10]. Thus, cold stamping is not suitable for the manufacture of complex structure. The punches and dies also need constant maintenance or replacement due to material scratching, scoring and galling caused by large press capacities [11].

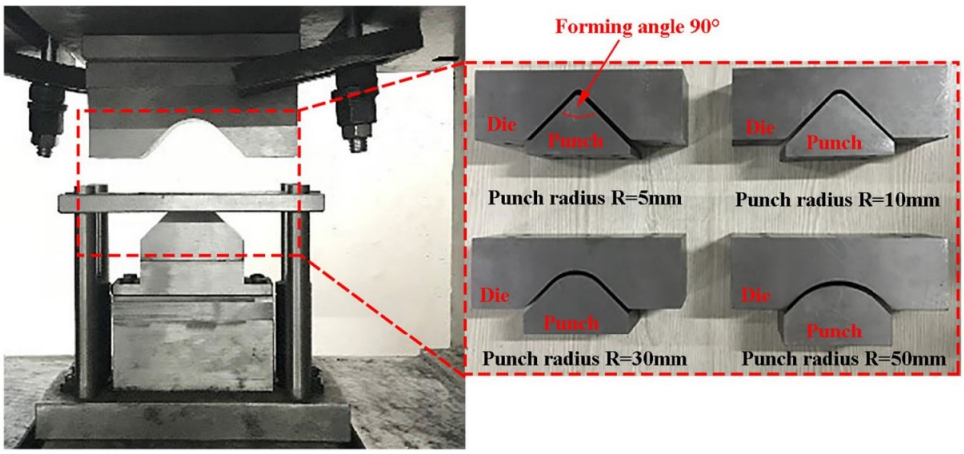


Figure 1-2. The V-shaped stamping tools in a cold stamping process [7].

Figure 1-3 presents the two main variants of the contemporary hot stamping process for steels [12], namely direct and indirect hot stamping method. In the direct hot stamping process, the sheet blank is first heated to austenitisation in a furnace at the temperature of about 900-950 °C [13]. After the blank is austenitised for 5-10 min to homogeneous austenitic microstructure, it is quickly transferred to cold dies, within a few seconds, for forming and quenching. In the indirect hot stamping process, the blank is first cold formed to a nearly completely formed shape before heated in the furnace. After austenitisation, it is transferred to the press for calibration and quenching. These two processes can also be used for other metals such as aluminium alloys.

Through heating, the formability of aluminium alloys at elevated temperature can be significantly improved, especially for high-strength alloys like AA7075 [7]. Thus, the increased ductility enables the production of more complex-shaped components and extends the aluminium alloy applications to transportation industry. The combination of hot forming and quenching remarkably reduces the springback effect, which contributes to much higher part dimensional accuracy than cold stamping [4]. However, with the introduction of heating source, hot stamping process requires more expensive equipment and longer production cycle [14]. In addition, it is less environmentally friendly than cold stamping.

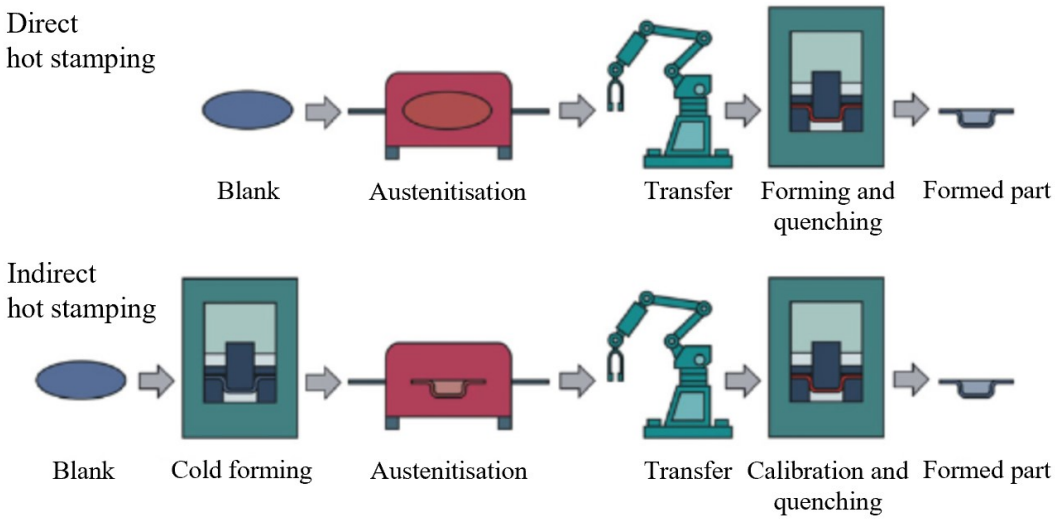


Figure 1-3. Two basic hot stamping processes: direct and indirect hot stamping [12].

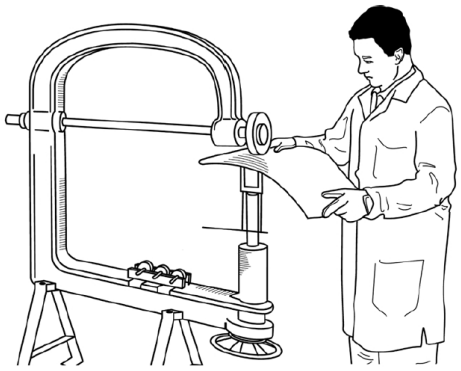
1.2 An alternative approach: free-form flexible sheet metal stamping

The formed parts from sheet metal stamping are subject to the unalterable shapes of punch and dies, for which the limited forming flexibility impedes the applicability of off-the-shelf stamping equipment to new sheet metal components. As reviewed by Trzepieciński [15], production flexibility in the development of contemporary sheet metal forming machines is becoming increasingly important. In addition, the extraordinarily high capital cost on customised punch and dies [16], especially for large-scale stamping, leads to expensive prototyping and arduous research and development of novel sheet metal designs. Thus, an alternative approach, flexible forming, is brought up to replicate the workflow of a blacksmith [17]. In traditional metal smithing, skilled workers can form parts of different geometries using limited number of general-purpose tools. Due to its labour-intensive and time-consumptive nature, it is mainly applied to personalised production.

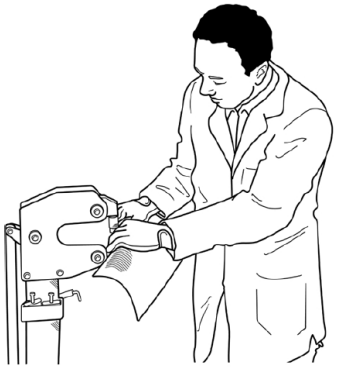
There are four most commonly used tools on the shop floor [18], namely hammer, Kraftformer, English wheel and spinning lathe, as shown in Figure 1-4. During forming, the metal smith often uses a combination of these tools to form a single part. At hammering, the smith rests the sheet metal on an anvil, such as wooden dollies and sandbags, and repeatedly strikes the sheet with a hammer in his other hand. The main forming processes during hammering include folding, hollowing, raising and planishing. For English wheel, the sheet metal is rested between two anvil rollers, which are mounted between the jaws of a C-shaped frame. The upper and lower roller have a flat and crowned profile, respectively, and the sheet is manoeuvred back and forth in a 'zig-zag' pattern to render a compound curvature. A Kraftformer is a machine housing a reciprocating press which can stretch and shrink the sheet metal. During forming, the sheet is incrementally deformed to its target shape through the repeated deformations. A spinning lathe is used to form axisymmetric parts, of which a mandrel is clamped to the headstock of a lathe and the sheet blank is clamped in-line by a tailstock. The lathe is actuated for spinning, and the smith pokes the sheet metal, using a forming tool, at near the centre of the blank and travels to its outer perimeter and back.



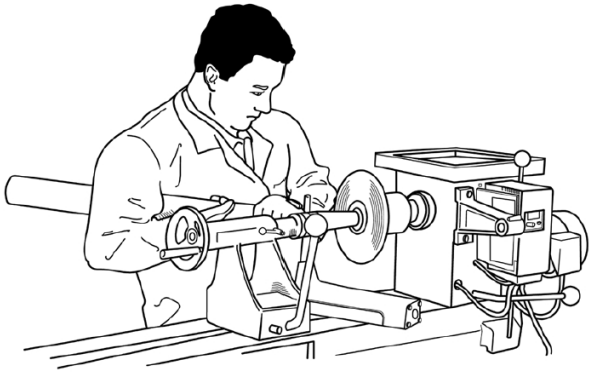
(a) Hammer



(b) English wheel



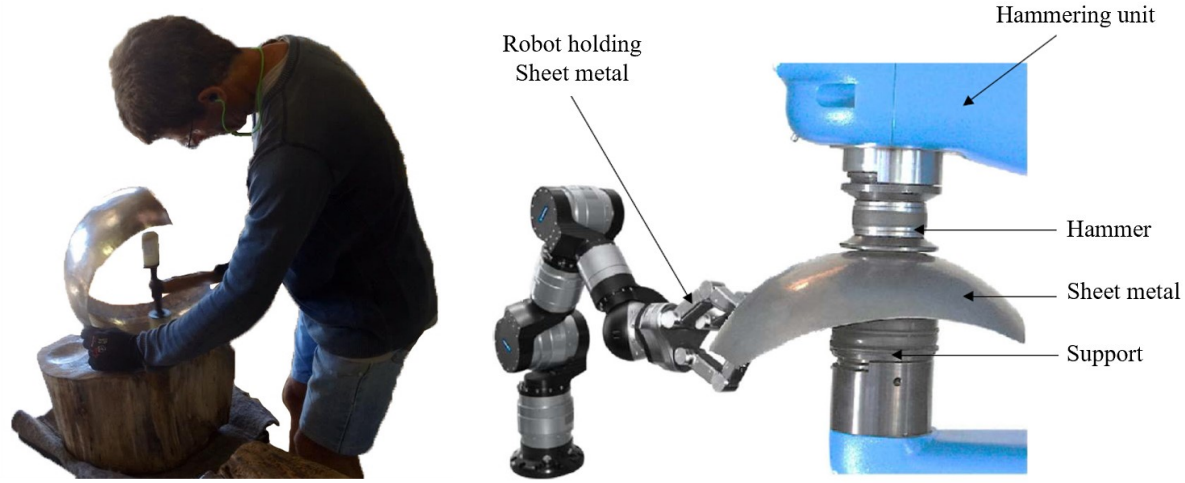
(c) Kraftformer



(d) Spinning lathe

Figure 1-4. Four commonly used tools by the metal smith on the shop floor [18]**.**

To increase the production efficiency and reduce the labour cost of traditional flexible forming to meet the requirement of high-volume personalised production in sheet metal forming industry nowadays, the research on flexible forming over the last two decades has been focusing on automating traditional techniques using bespoke machinery or industrial robot arms. Not only were the traditional smithing techniques replicated, the smith's heuristics/tacit knowledge can be also embedded to optimise the forming strategy. Figure 1-5 shows examples of both manual and automated hammering processes, of which the human manipulation in the former is replaced by an industrial robot arm in the latter. Most of the automated processing strategies are found to be generated from a knowledge-based (KBS) or rule-based system, which encodes previous practical tool path planning experiences or cases into forming rules and stored in a knowledge base. In a KBS, three reasoning schemes are commonly used for decision-making [19]: rule-based reasoning (RBR), case-based reasoning (CBR) and model-based reasoning (MBR). For RBR, rules of the forming process are concluded in an IF-THEN format and used for tool path reasoning [20]. For CBR, previous experience of a similar forming condition is used to yield a decision [21]. For MBR, a model is developed for modelling the part and forming operations, with which the physical forming process is modelled and compared with the observed data to yield a decision [22]. However, as the KBS formulate the forming decision in accordance with the pre-defined rules or previous forming experience in the knowledge base, it cannot formulate better decisions than existing strategies in the knowledge base. In a word, the KBS cannot self-improve its decision-making capability unless the knowledge base is enriched with new rules by human experts.



(a) Manual hammering [23] **(b) Automated hammering** [24]

Figure 1-5. Manual and automated sheet metal hammering processes.

1.3 Aims and approach

Over the last two decades, machine learning technology has made its well-known breakthrough in Computer Vision (CV) and Natural Language Processing (NLP) for its remarkable feature learning capability in both spatial and temporal dimensions. Although machine learning has been vastly applied to some areas of sheet metal production, such as fault diagnosis and surrogate modelling, only a few applications are found for the development of sheet metal tool path planning system. Thus, this gap motivates the research on developing intelligent flexible sheet metal forming techniques by applying the state-of-the-art machine learning technologies to flexible free-form processes.

The main aim of this thesis is to develop machine learning based tool path planning techniques for an intelligent sheet metal forming system, which is capable of self-improving its tool path planning strategies without human intervention. In addition, methods for alleviating the problem of expensive acquisition of computational or experimental data are also to be attempted.

To achieve the aims, two out of three principal machine learning techniques, namely supervised learning (SL) and reinforcement learning (RL), are investigated in this thesis. The detailed approaches used in this research are listed as follows:

- Review the essential research literature on conventional sheet metal stamping processes, flexible sheet metal free forming processes, machine learning theories and their applications to sheet metal stamping processes, especially applications to tool path planning strategies.
- Propose an overall architecture of the intelligent flexible sheet metal forming system, with modular design for parallel operation, control and development of different functions of the system.
- Develop a theory-guided machine learning technique which incorporates theories in mechanics of materials to improve model interpretability and alleviate the demand for big data.
- Propose a recursive tool path prediction framework for a novel flexible free-form sheet metal stamping process and investigate the tool path planning performance of state-of-the-art deep learning models for this process.

- Develop and verify a generalisable tool path planning strategy, which can be self-improved without human intervention, applicable for any arbitrary target sheet metal shape.

1.4 Thesis structure

Figure 1-6 presents the flowchart of the thesis structure.

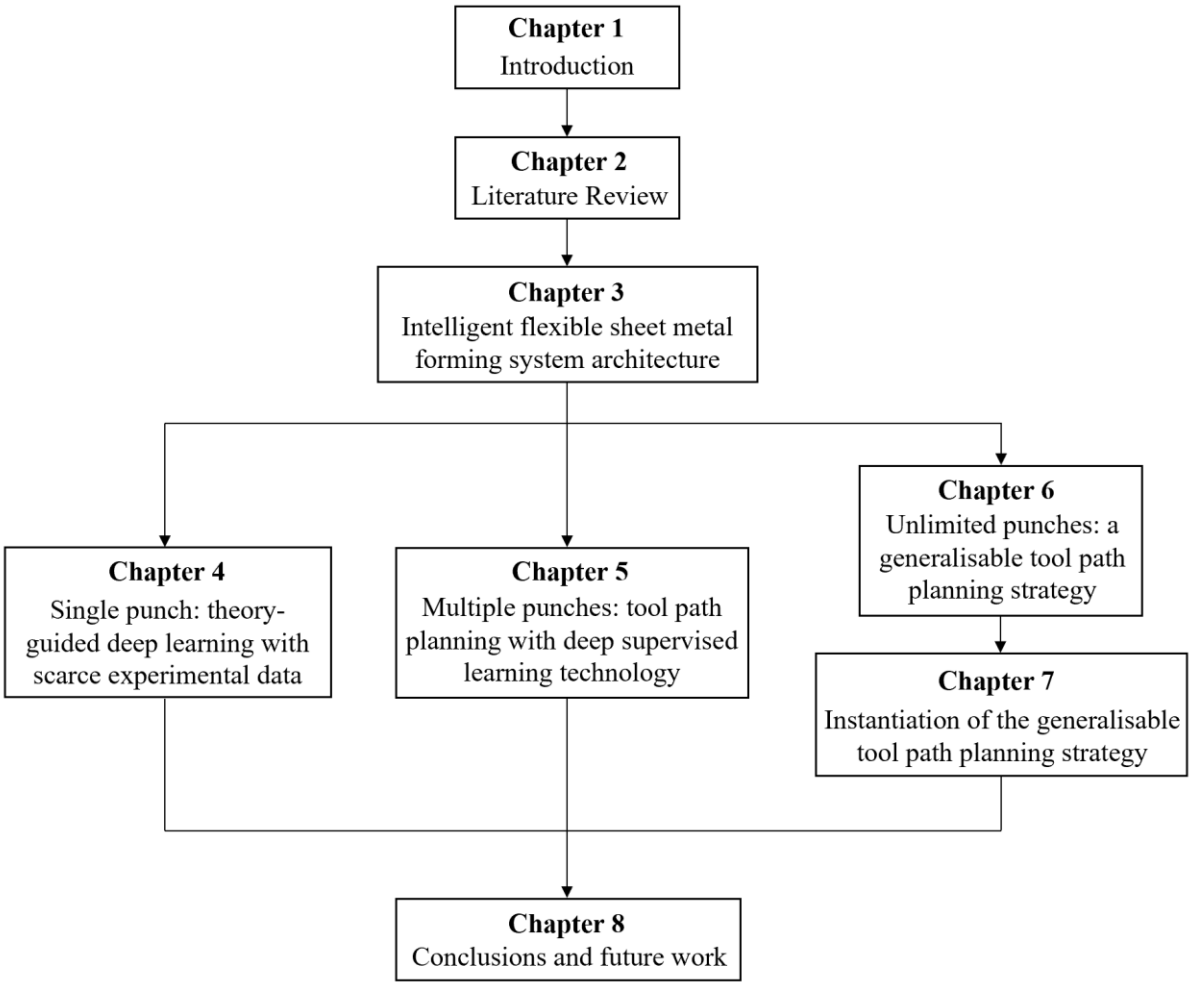


Figure 1-6. Flowchart of the thesis structure.

Chapter 1 introduces the background of contemporary sheet metal stamping technology, followed by the necessity of developing automated free-form flexible forming techniques and the motivation of process intellectualisation. The aims of this thesis and relevant approaches are also introduced.

Chapter 2 comprehensively reviews relevant literatures to this research, including conventional sheet metal stamping processes, recent development of an automated flexible forming process, theories and applications of the state-of-the-art machine learning technologies and an exhaustive investigation of machine learning applications to sheet metal stamping process.

Chapter 3 proposes the overall architecture of the intelligent sheet metal forming system for tool path planning, which is composed of seven modules responsible for data processing and storage, tool path planning, process computation/validation and feedback control.

Chapter 4 develops a novel theory-guided deep learning method, through multi-task learning and parameter norm regularisation with Swift's law. This method is compared with pure data-driven method under various experimental conditions.

Chapter 5 develops a recursive tool path prediction framework for a consecutive rubber-tool forming process, of which deep learning models play the dominant role. Three series of state-of-the-art deep learning models are compared in terms of their tool path learning performances.

Chapter 6 investigates the potential of harnessing reinforcement learning technology for tool path planning and, from the investigation results, proposes a generalisable strategy for arbitrary sheet metal parts.

Chapter 7 instantiates the generalisable tool path planning strategy proposed in Chapter 6, using computational forming results. Six state-of-the-art deep reinforcement learning algorithms and three deep neural networks are implemented, followed by a comprehensive discussion on the learning performance and characteristics of the generalisable strategy.

Chapter 8 concludes the main outcomes and contributions of this thesis in developing intelligent metal forming system. Potential future work to improve system performance and promote applications is also suggested.

Chapter 2

Literature Review

This chapter first reviews conventional sheet metal stamping processes in Section 2.1, which have been intensively developed for decades and used for mass production of lightweight panels. Then, a flexible sheet metal free forming process, which has higher forming flexibility than conventional stamping, is reviewed in Section 2.2. Section 2.3 and 2.4 introduce the mathematical physics behind deep supervised learning and deep reinforcement learning technologies, respectively, in which their prominent applications are also summarised. Section 2.5 comprehensively reviews machine learning applications to sheet metal stamping processes, covering process monitoring, fault diagnosis, surrogate assisted optimisation and tool path planning. Finally, A summary is given in Section 2.6.

2.1 Conventional sheet metal stamping

As introduced in Section 1.1, sheet metal stamping is the most widely used forming technology for sheet panels, whose applications have permeated into automobile and aircraft industry. This section reviews the stamping process and one of the most common stamping defects, springback.

2.1.1 Cold, warm and hot stamping

Sheet metal stamping techniques can be classified as cold stamping and hot stamping, based on whether a furnace is included to heat the sheet metal. As shown in Figure 2-1, hot stamping can be further divided into two groups, warm stamping and hot stamping, with warm forming under recrystallisation temperature (340 to 400 ℃ for aluminium alloy [25]) and hot forming above.

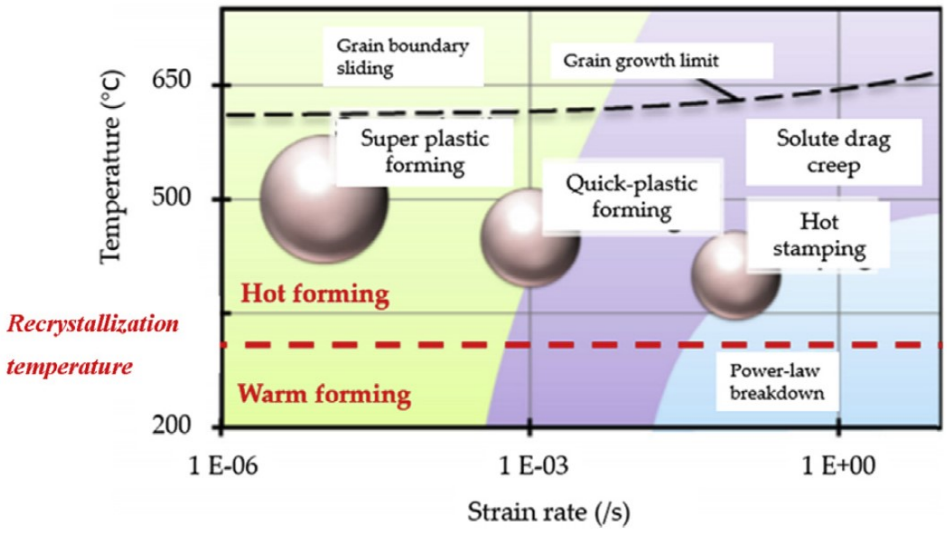


Figure 2-1. Forming temperatures for different forming processes [4].

The cold stamping process has been introduced in Section 1.1, the sheet metal is first clamped by blank holders at edges and then stamped by punch and die. This process is most commonly used for mass production in industry because of its simple setup and high production efficiency. However, cold stamping can leads to significant sheet metal springback due to the elastic stress relaxation after the punch load is removed, causing the loss of dimensional accuracy [5]. The characteristics of springback will be presented in Section 2.1.2. In addition to springback, fractures can also occur due to the limited ductility of aluminium alloy at room temperature. To improve ductility, a discreet selection of heat treatment temper for the initial blank is required, such as T4 and T6 [26]. Figure 2-2 shows an example of the effect of temper on ductility, in which an evident increase of ductility of AA6111 with W temper can be observed compared to that with T6 temper.

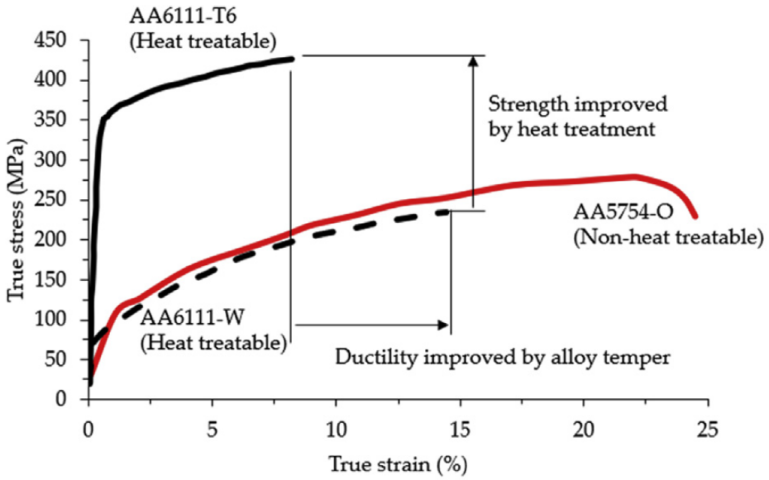


Figure 2-2. Stress-strain curves of aluminium alloys with different tempers [4].

To improve sheet metal formability and alleviate springback, the sheet metal can be heated and stamped at elevated temperature. Two basic warm/hot stamping processes have been shown in Figure 1-3, and an example of warm stamping experimental setup is presented in Figure 2-3. In this process, the aluminium sheet was heated before clamped by the blank holder and drawing die. The blank holder was also heated using a built-in electric heater, while the punch was cooled by built-in water-cooling channels. The non-uniform temperature distribution caused by the cooled punch was reported to increase the formability of sheet metal [27].

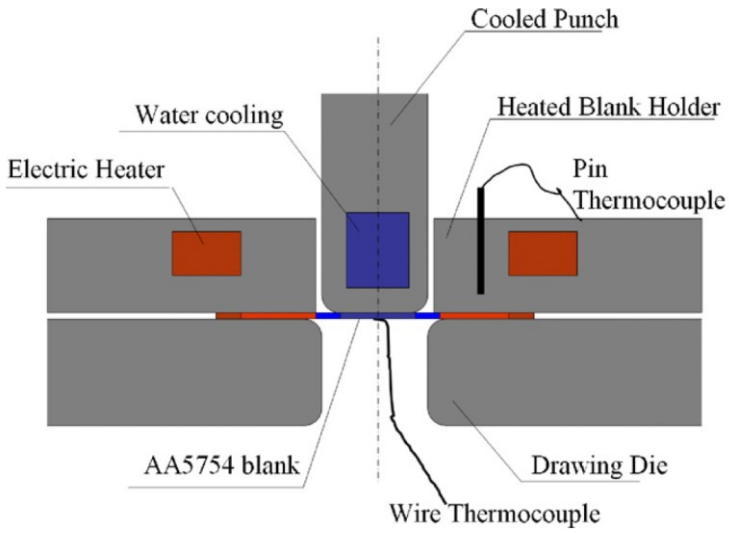
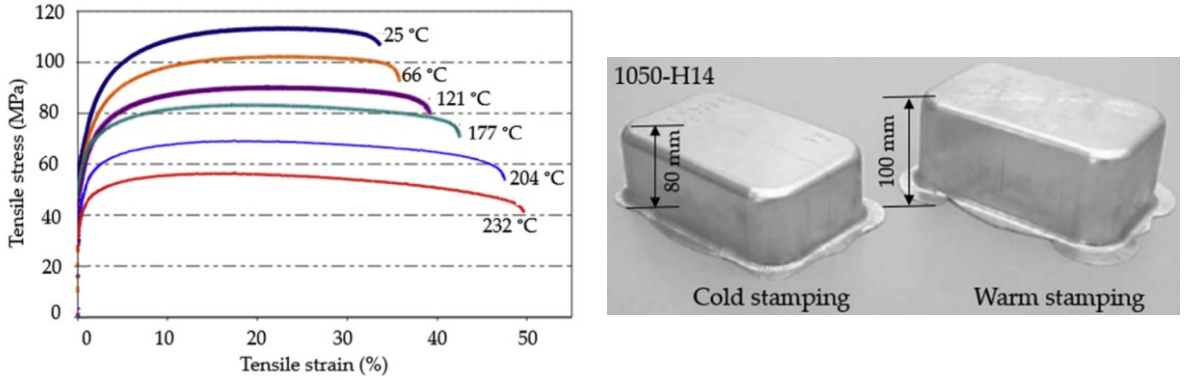


Figure 2-3. Experimental setup of warm stamping [28]**.**

Figure 2-4 shows the effect of forming temperature on stamping. In Figure 2-4a, there is an evident trend that the fracture strain increases with temperature and the tensile stress for the same strain decreases with temperature, which indicates that the aluminium alloy AA3003-H111 ductility increases with stamping temperature. The fracture strain of this aluminium alloy in 232 ℃ warm stamping increased over 50% compared to that in cold stamping (25 ℃**).** Figure 2-4b also demonstrates that warm stamping leads to better drawability of AA1050-H14 than cold stamping, in which an extra 20 mm of drawing depth was achieved in warm stamping. In addition to heightened ductility, reduced springback and improved dimensional accuracy were also brought by warm stamping [29].



(a) Stress-strain curves [30]. **(b) Cold and warm stamped parts** [31].

Figure 2-4. Effect of forming temperature on stamping performance.

In hot stamping, the sheet metal is heated to a temperature above crystallisation temperature. Compared to warm stamping which is not suitable for high-strength heat treatable aluminium alloys [32], higher forming temperatures makes the hot stamping of high-strength aluminium alloys feasible and efficient. A canonical hot stamping process, namely Hot Form and Quench (HFQ®), was developed by Lin et al. [33] to integrate heat treatment, hot forming and quenching.

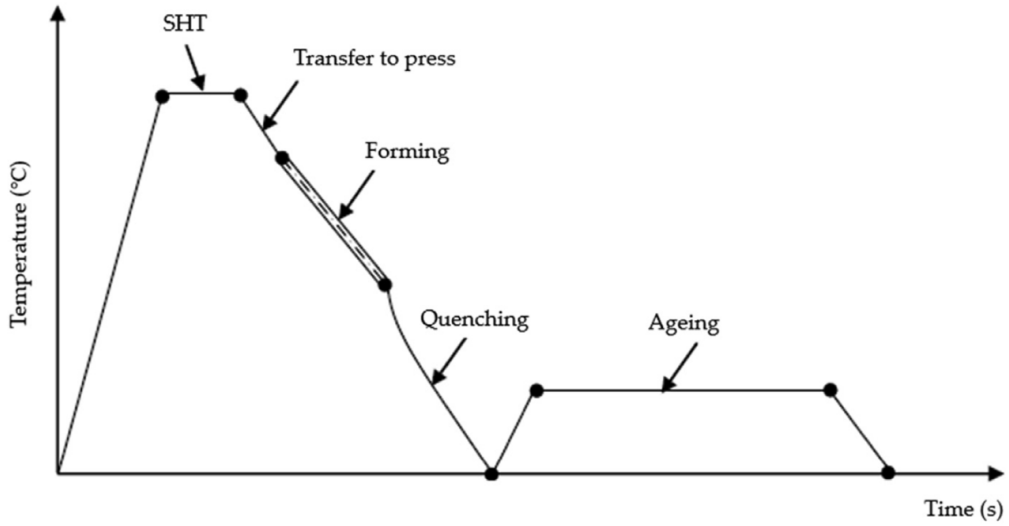


Figure 2-5. Sheet metal temperature profile during HFQ® process [4].

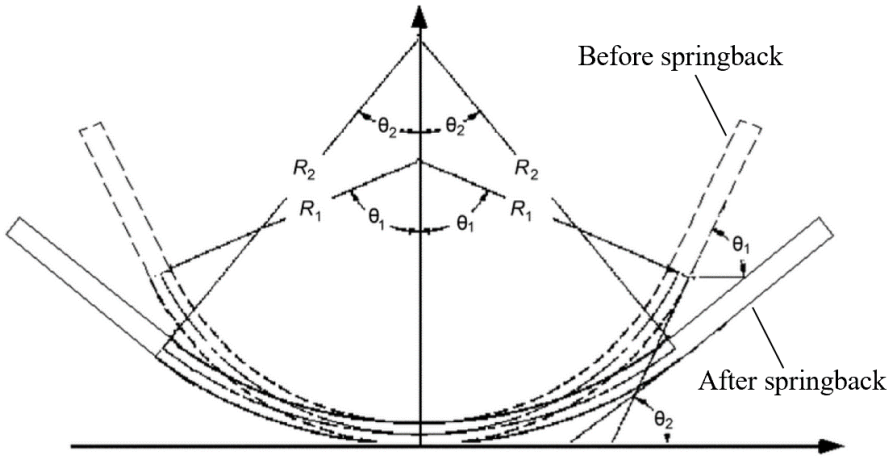
The HFQ® process is briefly described with a temperature profile in Figure 2-5, in which the sheet blank is first heated to a solution heat treatment (SHT) temperature (450 to 600°C) and maintained in the furnace for 20-60 mins to obtain an optimal microstructure. The blank is then rapidly transferred to cold dies within 5 s for stamping and quenching for about 30 s. The quenched part is finally artificially aged to allow precipitation hardening and strength enhancement. HFQ® was reported to produce optimal microstructure, maximise aluminium

alloy ductility, improve productivity, eliminate springback and ensure dimensional accuracy [4].

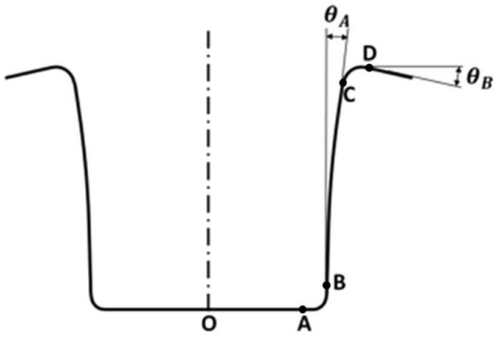
Although stamping technologies have been developed and immensely exploited in sheet metal forming industry, it needs adaptations for more flexible forming processes to fulfil the increasing demand on personalised production nowadays. The research on flexible forming will be reviewed in Section 2.2.

2.1.2 Springback

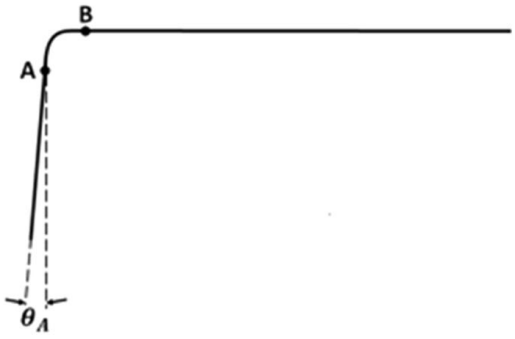
Springback is one of the most common effects occurring in sheet metal forming processes, which is caused by the elastic recovery of materials during unloading. This effect exacerbates when stamping high strength aluminium alloys or steels which have high yield strength-to-elastic modulus ratio. Due to this effect, the stamped products can have significant dimensional deficiency.



(a) Air bending [34]



(b) U-shape bending [9]



(c) L-shape bending [9]

Figure 2-6. Springback angle notations for different bending processes.

Figure 2-6 presents the springback angles for three bending processes, which are commonly used to measure the dimensional difference of the sheet metal before and after unloading. For air bending, the springback angle is $2(\theta_1 - \theta_2)$. For U-shape and L-shape bending, θ_A and θ_B are used for the former and θ_A is used for the latter.

Figure 2-7 shows two examples of springback occurred in practical U-shape bending and L-shape bending process, respectively.



(b) U-shape bending [35] **(c) L-shape bending** [9]

Figure 2-7. Springback in practice for different bending processes.

To surmount the springback in sheet metal forming process, most of the current research focuses on developing compensation methods in numerical simulations or experiments [36,37]. For instance, Lingbeek et al. [38] developed the smooth displacement adjustment (SDA) method and the surface controlled overbending (SCO) method to optimise the tool shape for a deep drawing process to increase the geometry accuracy of the product after springback, with finite element (FE) simulations. Wang et al. [39] developed a three-dimensional (3D) scanning method to replace the traditional special fixture for inspecting the part during the stamping of a rear reinforcement plate of a door panel, which was stated to increase the qualified rate of the product to a new peak. In creep age forming (CAF), Li et al. [40] proposed a one-step springback compensation method for singly curved products and an accelerated method for CAF tool design with reference to the springback compensation curves, which were established based on the numerical solution of springback behaviour of CAF process. The effectiveness of the two methods was demonstrated by CAF experimental tests.

Although springback occurring in different sheet metal forming processes has been significantly alleviated by the increasing amount of research in this area, the development cycle is still considerably time-consumptive and demands high computational or capital cost. In this context, the impressive learning capability and low temporal cost of machine learning

technology has attracted researchers to harness it in sheet metal forming industry, which will be reviewed in Section 2.5.

2.2 Flexible sheet metal free forming – automated hammering

To extricate metal smiths from laborious smithing work and improve production efficiency, the traditional handcrafting workflow is automated through bespoke machinery or industrial robot arms and a tool control system. During the last two decades, research has been found on automating the four most commonly used forming operations on the shop floor [18], namely hammering, wheeling, driving (Kraftformer) and spinning. Since this thesis focus on developing an intelligent system for a novel hammering process, only the literatures of automated hammering are reviewed in this section.

Most of the research on automated hammering system was pioneered in Japan, which reserved the key components in a traditional hammering process, namely the punch and anvil. An early automated hammering system was realised, by Mori et al. [41], through actuating a hemispherical punch using a hydraulic unit and striking an aluminium square sheet rested on a flat die, as shown in Figure 2-8. The sheet was modelled through a number of scattered points at equal spaces, at which the impact load was applied. The hammer deformed the sheet with a constant load, and the hammering sequence (tool path) was optimised using genetic algorithm (GA). Due to the massive data required for GA optimisation, empirical equations were used to approximate the local curvature after the blow, with the curvature of the punch location and its four neighbouring points before the blow. However, the experiments showed that there is an evident difference between the GA-obtained and desired curvature distribution, which was reported to be caused by the determination of coefficients in the empirical equations. To improve the hammering sequence strategy, Mori developed a KBS with a case-based reasoning scheme, which forms the sheet metal by roughing and finishing operations [42]. Pre-defined geometries with their tool paths were stored in the database, and a similar geometry to the target was found by a pattern recognition approach. The forming results had a good agreement with the desired shape, except for those with sharp corners.

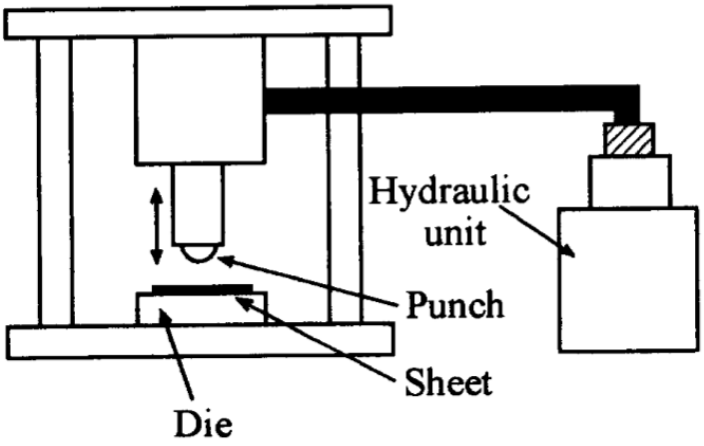
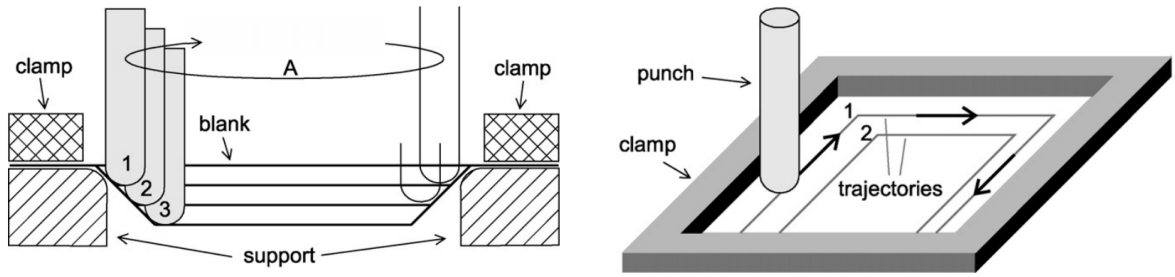


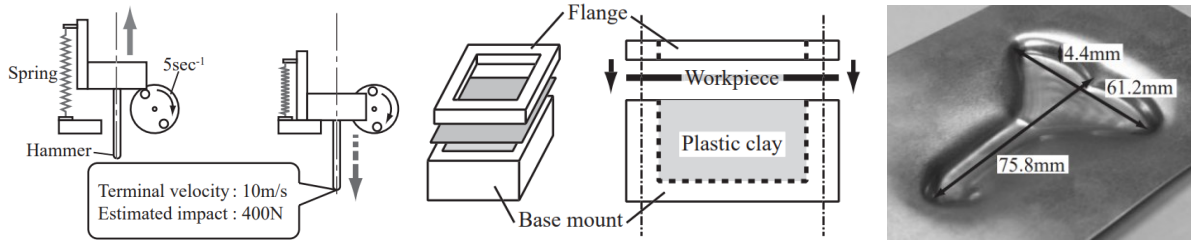
Figure 2-8. The automated hammering system for sheet metals [41].

Another automated hammering process, named incremental sheet punching (ISP), is very similar to the well-known incremental sheet forming (ISF) process. Figure 2-9 shows a typical ISF process, of which the sheet metal is clamped at edges and deformed by continuously moving a round tipped tool and applying localised deformations [5]. By contrast, ISP deforms the sheet metal in a similar continuous pattern but with a reciprocating punch. An early automated ISP process was developed by Tanaka et al. [43], as shown in Figure 2-10. A servo hammering unit was mounted to a spindle, whose rotation would drive the hammer to reciprocate. The workpiece was clamped on a base mount (die) by a flange (blank holder), and the base is filled by plastic clay for elastic support. The hammer struck the workpiece with a constant impact force, and the tool path for the hammer movement was manually generated based on the target workpiece CAD, tool shape, crossfeed, depth and tool path mode. Although the hammered workpiece, as shown in Figure 2-10c, had a dimensional error of about 12 % in thickness direction, the successful automation of the hammering process demonstrated the system potential.



(a) ISF process experimental setup. **(b) ISF process trajectories.**

Figure 2-9. A single-point incremental sheet forming process [44].



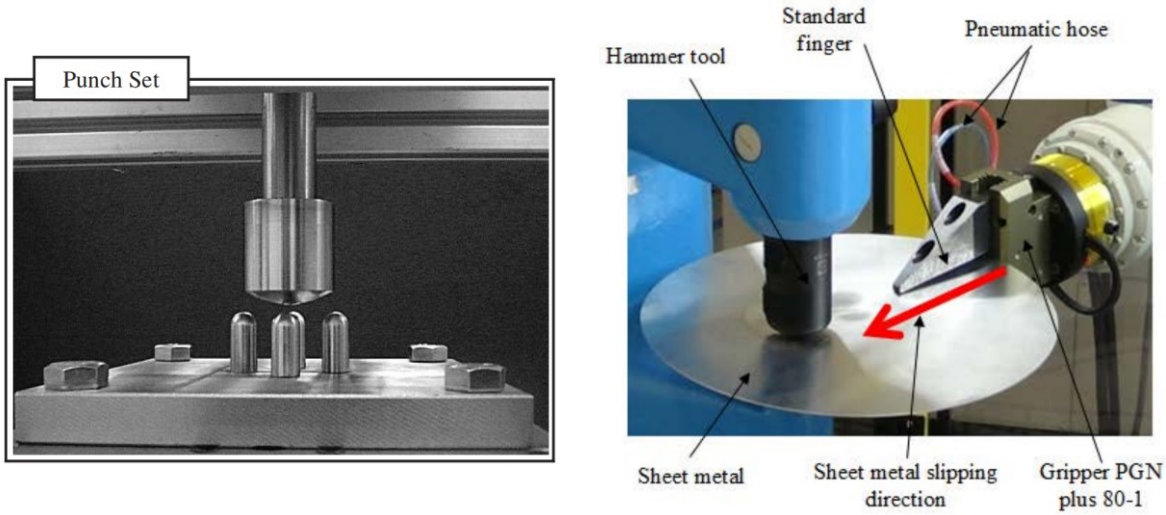
(a) Mechanised hammer **(b) Workpiece and anvil** **(c) Hammered workpiece**

Figure 2-10. An automated ISP process with its (a) hammering equipment, (b) forming setup and (c) forming result [43].

Tanaka et al. [45] continued their research on automated ISP and optimised the hammering mode, such as hammering angles, to compensate the dimensional error of hammered workpiece in workpiece depth, cylindrical corners and spherical corners. The hammer was mounted to a 6-DOF (degree of freedom) industrial positioning robot. To prevent the workpiece from fracture or breakage during ISP process, Asakawa and Tanaka [46] developed a tool path generation method using parametric curve interpolation, which considered continuity to generate tool paths with a gradual and layer-by-layer change. After then, they [47] investigated the relationship between the hammering direction and its effect on production shape, based on which a method of tool path generation using non-uniformed rational B-spline (NURBS) expression was proposed. This method was instantiated by forming a complex-shape workpiece, and the hammered part had a high dimensional accuracy except for the part near the vertical wall. To further improve the forming accuracy and system control, Tanaka et al. [48] constructed a feedback control system by measuring the shape of workpiece in the hammering process. Similar automated ISP systems were also developed by Schafer and Schraft [49], Luo et al. [50] and Wang et al. [51].

Apart from automating traditional hammering process with a punch and die, new hammering techniques were also developed and automated. Yoon and Yang [52] proposed a new sheet metal forming process, of which the sheet metal was rested on two pairs of adjustable punch-shaped dies and struck by a centre punch, as shown in Figure 2-11a. The general forming methods for this hammering process were defined, and an analytical model for the curvature prediction of the formed sheet metal was developed using the test setup topology and process variables. However, the model cannot attain a good prediction for deformations with large springback. Another new automated hammering process was developed by Ilangovan et al. [24], which was automated and controlled through a mechatronics-based monitoring and control system. The sheet metal, rested on a support, was manipulated by a robotic arm for positioning

as shown in Figure 2-11b. The best tool path was manually designed through finite element (FE) modelling and experiments, and the rules of manipulation for the sheet metal during hammering were derived from human skills data captured and interpreted by two Vicon T-series 2 megapixel cameras. However, this system had not yet been evaluated concerning its forming accuracy and tool path efficiency on a prototype with FE simulations or experiments.



(a) Centre punch and adjustable dies [52] **(b) Fixtureless hammering** [53]

Figure 2-11. New automated hammering process with (a) centre punch and adjustable dies and (b) fixtureless manipulation.

2.3 Machine learning: supervised learning and applications

Over the last two decades, machine learning technology, especially deep learning, has seen its resurgence and burgeoning development in various areas thanks to the revolutionarily increased computation power of computer processors. Mainly driven by probability theories, machine learning teaches machine how to learn the intrinsic function of given data. A simplistic solving process of a practical application can be summarised as follow [54]: a set of given data (named *training set*) is first trained to determine the machine learning algorithm in this application which can be expressed as a function $y(x)$, a target vector t is used to express the category of each input x (each input has a unique target vector). Thereafter, this algorithm is used to take a new input x (named *test set*) and generate an output y which is encoded in the same way as the target vectors. The ability of this machine learning algorithm to correctly categorise new input from test set which differs from those used in training set is known as *generalisation*.

Since the target vector plays a significant role in the learning process, three widely recognised machine learning techniques were classified as follow [55]:

- *Supervised learning*: the training data consist of both input vectors and their corresponding target vectors.
- *Unsupervised learning*: only input vectors but no target vectors are included in training data.
- *Reinforcement learning*: in this technique, since there are no provided examples of optimal outputs to the learning algorithm, the maximum rewards of outputs are instead discovered by a process of trial and error.

In supervised learning, a parametric model is used to fit the training data. The model can be trained in a data-driven approach or theory-guided approach, depending on whether application-relevant theories are used to assist the learning process.

2.3.1 Data-driven learning

Like classical optimisation algorithms, machine learning algorithms solve tasks by searching a global minimum of an objective function, which is normally composed to measure the deviation of model predictions from their targets. The commonly used objective functions in machine learning are negative log-likelihood (NLL) and mean square error (MSE), which are used to solve two most common tasks, namely classification and regression. The function of NLL is derived from the Kullback–Leibler (KL) divergence, which measures the statistical distance between the probability distributions from the training data and the model [56]:

$$D_{KL}(p_{data} \parallel p_{model}) = \frac{1}{k}\sum_{i=1}^{k}\left[\log\left(p_{data}(\boldsymbol{x}_i)\right) - \log\left(p_{model}(\boldsymbol{x}_i;\boldsymbol{\theta})\right)\right] \qquad (2\text{-}1)$$

where p_{data} and p_{model} denote probability distribution from the training data and the model, respectively. k denotes the total number of data and $\boldsymbol{x}_i$ denotes an element of data. $\boldsymbol{\theta}$ denotes the model parameters.

Thus, the objective of classification task is to minimise the KL divergence. Since the data probability distribution only depends on the training data generation (constant), the objective function can be simplified to NLL function:

$$J_{NLL}(\boldsymbol{\theta}) = -\frac{1}{k}\sum_{i=1}^{k}\log\left(p_{model}(\boldsymbol{x}_i;\boldsymbol{\theta})\right) \tag{2-2}$$

For regression task, the objective function is composed to measure the Euclidean distance between prediction from the model and the target [56]:

$$J_{MSE}(\boldsymbol{\theta}) = \frac{1}{m}\sum_{j=1}^{m}\left(y_j(\boldsymbol{X}) - \hat{y}_j(\boldsymbol{X};\boldsymbol{\theta})\right)^2 \tag{2-3}$$

where m denotes number of tasks. $y_j(\boldsymbol{X})$ denotes the target of task j and $\hat{y}_j(\boldsymbol{X};\boldsymbol{\theta})$ denotes the prediction of task j from the model.

Gradient descent algorithms are commonly used in machine learning for optimisation, which differentiate the objective function with respect to model parameters (variables) and iteratively search the global minimum in the gradient descent direction. Chain rule is used to factorise and compute the differentiation of objective function with respect to each variable. Stochastic gradient descent (SGD) is the most basic while most widely used algorithm in machine learning, which is defined as [56]:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta\nabla_{\boldsymbol{\theta}}J(\boldsymbol{\theta}) = \boldsymbol{\theta} - \eta\frac{1}{k'}\nabla_{\boldsymbol{\theta}}\sum_{i=1}^{k'}L(\boldsymbol{x}_i, y_i, \boldsymbol{\theta}) \tag{2-4}$$

where η denotes learning rate, k' denotes the total number of stochastic data (minibatch size) which are uniformly sampled from the training dataset, and $L(\boldsymbol{x}_i, y_i, \boldsymbol{\theta})$ denotes per-example loss. The arrow denotes a single update step of model parameters, $\boldsymbol{\theta}$, from right hand side to left hand side. Through minibatch training, the problem of prohibitively expensive computational cost in batch training, especially with large training sets, is alleviated [56]. There are also several derivatives of SGD in recent years which were found to be training-efficient, such as Adam [57], AdaGrad [58] and RMSProp [59].

With the optimisation algorithm, the internal parameters, weight matrices and bias vectors, of machine learning models are changed by using backpropagation algorithm and chain rule. The goal is to iteratively update the model parameters to obtain the lowest possible objective function value before overfitting occurs. The optimisation problem can be defined as:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, J(\boldsymbol{\theta};\boldsymbol{X},\boldsymbol{y}) \tag{2-5}$$

where $\boldsymbol{\theta}^*$ denotes the optimal value of $\boldsymbol{\theta}$ to obtain the minimum value of the objective function at the end of optimisation. $\boldsymbol{X}$ and $\boldsymbol{y}$ denote input data and targets, respectively.

2.3.2 Theory-guided learning

The most prevailing method for theory-guided machine learning is to introduce a regularisation term, which could be derived from application-relevant theories, to the objective function. Regularisation methods have been applied to traditional machine learning for decades, which are proved to be also effective in limiting the capacity of machine learning models. Parameter norm regularisation introduces a parameter norm penalty, $\alpha\Omega(\boldsymbol{\theta})$, to the original objective function shown in Eq. (2-3) and obtain a regularised objective function [56]:

$$\hat{J}(\boldsymbol{\Theta}; \boldsymbol{X}, \boldsymbol{y}) = J(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) + \alpha\Omega(\boldsymbol{\theta}) \tag{2-6}$$

where $\alpha \in [0, \infty)$ is a hyperparameter used to control the relative contribution of the norm penalty term, $\Omega(\boldsymbol{\theta})$, regarding to the original objective function, $J(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y})$. $\boldsymbol{\Theta}$ represents all the parameters involved in this network, including weights, biases and hyperparameters. It can be seen that the penalty term is a function of trainable parameters, $\boldsymbol{\theta}$. By controlling the training modes of the network parameters, the overfitting issues in many research studies can be addressed or alleviated [56].

2.3.3 Deep neural network

There are various machine learning models or techniques developed for classification and regression tasks, such as linear regression, decision tree [60], random forest [61], support vector machines (SVM) [62], Naive Bayes [63], K-nearest neighbours (KNN) [64] and artificial neural network (ANN) [65]. Neural network has been the most prevailing machine learning model as of today, ever since its high demand on processing power is met. It attracted even more interest of research after deep neural networks (DNNs), networks which have deep layers of neurons, were revealed to be able to recognise high-level concepts [66] and learn representations of data with multiple levels of abstraction [67].

There are three widely used architectures of neural networks, namely multilayer perceptron (MLP), convolutional neural network (CNN) and recurrent neural network (RNN). An MLP is

typically composed of an input layer, hidden layers and an output layer, which are shown in Figure 2-12. In MLPs, the basic element of the neural network is neuron, which is defined as:

$$y = f(w_1x_1 + w_2x_2 + \cdots + w_kx_k + b) = f(\boldsymbol{w}\boldsymbol{x} + b) \tag{2-7}$$

where *f* denotes the activation function (or transfer function). *w* and *b* denote weight and bias, respectively. Activation function is used for output processing for different learning tasks, such as linear activation for regression and sigmoid activation for classification in the output layer. Rectified linear unit (ReLU) is a commonly used activation function in the input and hidden layers [68].

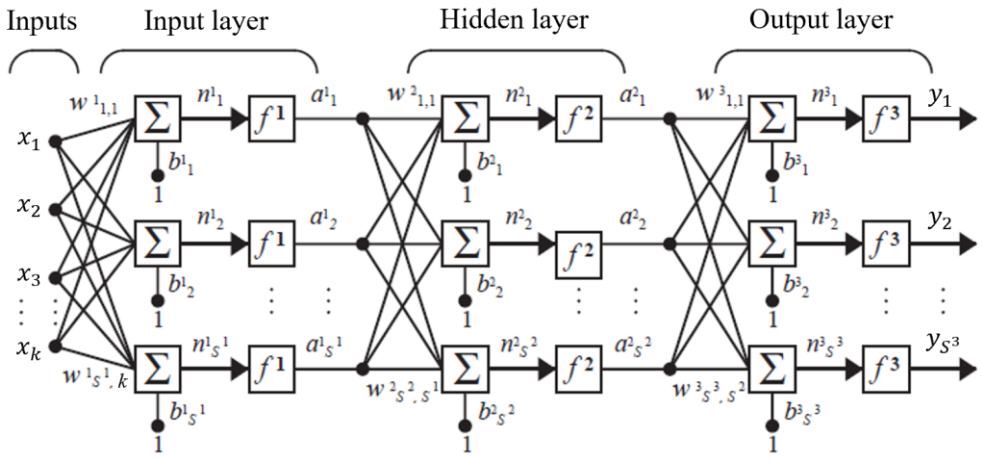


Figure 2-12. Three-layer fully connected neural network (adapted from [65]***).***

Thus, the three-layer MLP in Figure 2-12 is defined as:

$$\boldsymbol{y} = \boldsymbol{f}^3(\boldsymbol{W}^3\boldsymbol{f}^2(\boldsymbol{W}^2\boldsymbol{f}^1(\boldsymbol{W}^1\boldsymbol{x} + \boldsymbol{b}^1) + \boldsymbol{b}^2) + \boldsymbol{b}^3) \tag{2-8}$$

where the superscript denotes the layer number. For example, $\boldsymbol{f}^3$ denotes the activation functions in the third layer. $\boldsymbol{W}$ and $\boldsymbol{b}$ denote the weight matrix and bias vector, respectively.

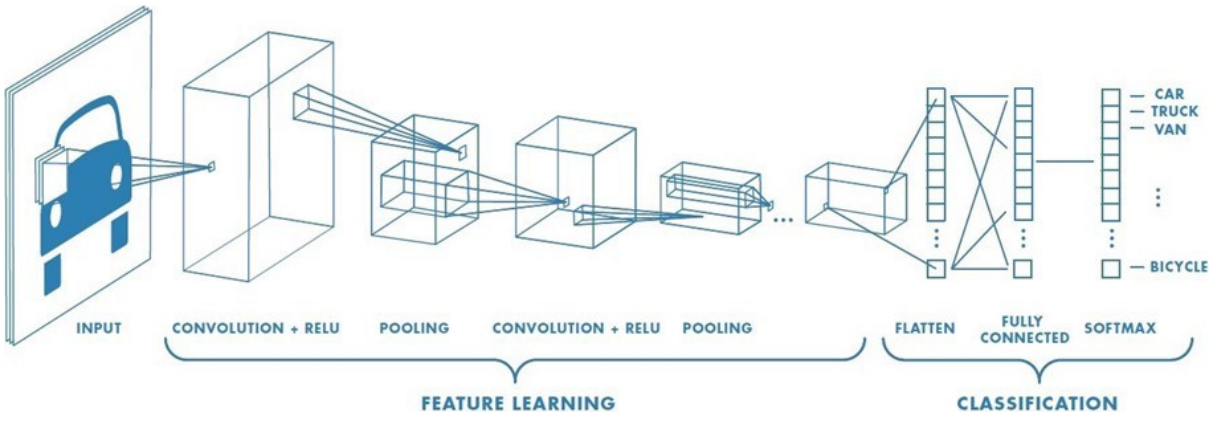


Figure 2-13. Architecture of a convolutional neural network [69]**.**

In 2012, a deep CNN [70] was found to achieve record-breaking results on a highly challenging dataset, ImageNet [71]. The study clarified the importance of the depth of CNN in improving network performance. Figure 2-13 shows the typical architecture of a CNN, which is composed of convolutional layers for feature learning and fully connected layers for classification.

The convolutional layer utilises the discrete convolution operation to shift the kernel function through the input space, which is defined as:

$$g^s(i^s) = (K^s * I^s)(i^s) = \left(\sum_{t=-\infty}^{\infty} I^s[i^s - t] \cdot K^s[t] \mid \forall\, i^s \in [1, m^s], \mathbb{Z}\right) \qquad (2\text{-}9)$$

where $g^s(\cdot)$ represents the convolution function, and the superscript "s" denotes the layer number. i^s denotes the i^{th} element of the output feature map at the s^{th} layer of network, and $\forall\, i^s \in [1, m^s], \mathbb{Z}$ denote that all i^s are integers bounded in the region of $[1, m^s]$. $K^s(\cdot)$ and $I^s(\cdot)$ represents the kernel function and the input at s^{th} layer, and the element inside the square bracket in the equation denotes the matrix index. The altering of t from $-\infty$ to ∞ signifies the shift of the kernel through the input space.

Thus, one single convolutional layer is then defined as:

$$\widehat{\boldsymbol{Y}}^{\boldsymbol{s}}_{conv.}(\boldsymbol{x^s}; \boldsymbol{\theta^s}) = (f(g^s_d(i^s) + b^s_d) \mid \forall\, i^s \in [1, m^s], \mathbb{Z}; \forall\, d \in [1, d^s], \mathbb{Z}) \qquad (2\text{-}10)$$

where d^s denotes the number of kernels used at the s^{th} convolutional layer. $\widehat{\boldsymbol{Y}}^s_{conv.} \in \mathbb{R}^{m^s \times d^s}$ is the output from the s^{th} layer, which represents the newly extracted feature map at this layer. f denotes the activation function, which is commonly ReLU.

The function of the 1-D max pooling layer is defined as:

$$\widehat{\boldsymbol{Y}}^s_{mPool}(\boldsymbol{x}^s) = \left(\max_{1 \le h \le H^s}\left[x_{(i^s-1)\cdot H^s + h}\right] \mid \forall\, i^s \in [1, m^s], \mathbb{Z}; \forall\, h \in \mathbb{Z}\right) \qquad (2\text{-}11)$$

where H^s denotes the height of the pooling at the s^{th} pooling layer, and $\max_{1 \le h \le H^s}\left[x_{(i^s-1)\cdot H^s + h}\right]$ yields the maximum value inside the pooling vector. The function of 2-D max pooling can be obtained by applying the 1-D function in two dimensions.

RNN is widely used to learn the temporal features of sequential data, such as text, audio and video. It is developed from the idea of a computational graph with repetitive structure, which unfolds a recurrent computation and shares the parameters across a deep network

structure [56]. A typical RNN has a successive neuron structure along a temporal sequence as shown in Figure 2-14, which exhibits temporal dynamic behaviour.

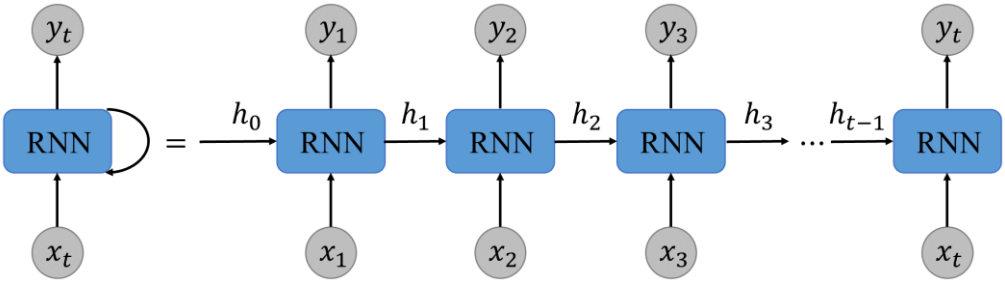


Figure 2-14. Architecture of RNN: rolled form (left) and unrolled form (right).

However, due to the vanishing gradient problem which can frequently occur during the training of RNNs, long short-term memory (LSTM) model [72] and gated recurrent unit (GRU) [73] were developed. In this thesis, only LSTM has been investigated. The vanishing gradient problem can incur data loss from previous time history and even impede the neural network from further training when the network becomes deep. Compared to vanilla RNN, LSTM brings in four new elements to cope with this problem: a cell, an input gate, and output gate and a forget gate. Figure 2-15 shows the architecture of a single LSTM unit.

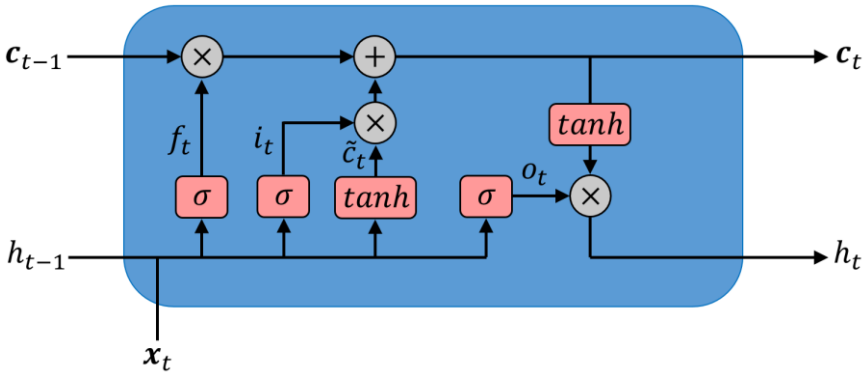


Figure 2-15. Architecture of LSTM.

The LSTM unit, developed by Hochreiter and Schmidhuber [72], is defined as:

$$
\begin{aligned}
f_t &= \sigma(\boldsymbol{W}_f \boldsymbol{x}_t + \boldsymbol{U}_f \boldsymbol{h}_{t-1} + \boldsymbol{b}_f) \\
i_t &= \sigma(\boldsymbol{W}_i \boldsymbol{x}_t + \boldsymbol{U}_i \boldsymbol{h}_{t-1} + \boldsymbol{b}_i) \\
o_t &= \sigma(\boldsymbol{W}_o \boldsymbol{x}_t + \boldsymbol{U}_o \boldsymbol{h}_{t-1} + \boldsymbol{b}_o) \\
\tilde{c}_t &= \tanh(\boldsymbol{W}_c \boldsymbol{x}_t + \boldsymbol{U}_c \boldsymbol{h}_{t-1} + \boldsymbol{b}_c) \\
\boldsymbol{c}_t &= f_t \odot \boldsymbol{c}_{t-1} + i_t \odot \tilde{c}_t \\
\boldsymbol{h}_t &= o_t \odot \tanh(\boldsymbol{c}_t)
\end{aligned}
\tag{2-12}
$$

where $\boldsymbol{x}_t \in \mathbb{R}^k$ is the input vector with dimension of k at t time step. $\boldsymbol{h}_t \in \mathbb{R}^h$ is the hidden state vector with dimension of h at t time step. f_t, i_t, o_t are the forget gate, input gate and output gate function, respectively. $\boldsymbol{c}_t \in \mathbb{R}^h$ is the cell state vector with dimension of h at t time step. $\boldsymbol{W} \in \mathbb{R}^{h \times k}$ and $\boldsymbol{U} \in \mathbb{R}^{h \times h}$ weight matrices. $\boldsymbol{b} \in \mathbb{R}^h$ is bias vector. σ and tanh denote the sigmoid and hyperbolic tangent activation function, respectively. $\odot$ denotes element-wise multiplication.

2.3.4 Applications

Most machine learning applications are purely data-driven, with regularisation methods, such as L1 and L2 [74], to prevent from overfitting and improve learning generalisation. Although this technology has been applied to innumerable fields in academia, it is widely recognised mainly for its remarkable learning performance in CV and NLP. In CV, machines are trained to gain high-level "understanding" from digital images or videos like human vision. There are four key machine learning application scenarios, namely image recognition [75], visual tracking [76], semantic segmentation [77] and image restoration [78]. In NLP, machines are trained to "understand" human languages and gain insights from textual/speech documents. The main NLP tasks include information extraction [79], machine translation [80], sentiment analysis [81], speech processing and recognition [82], etc.

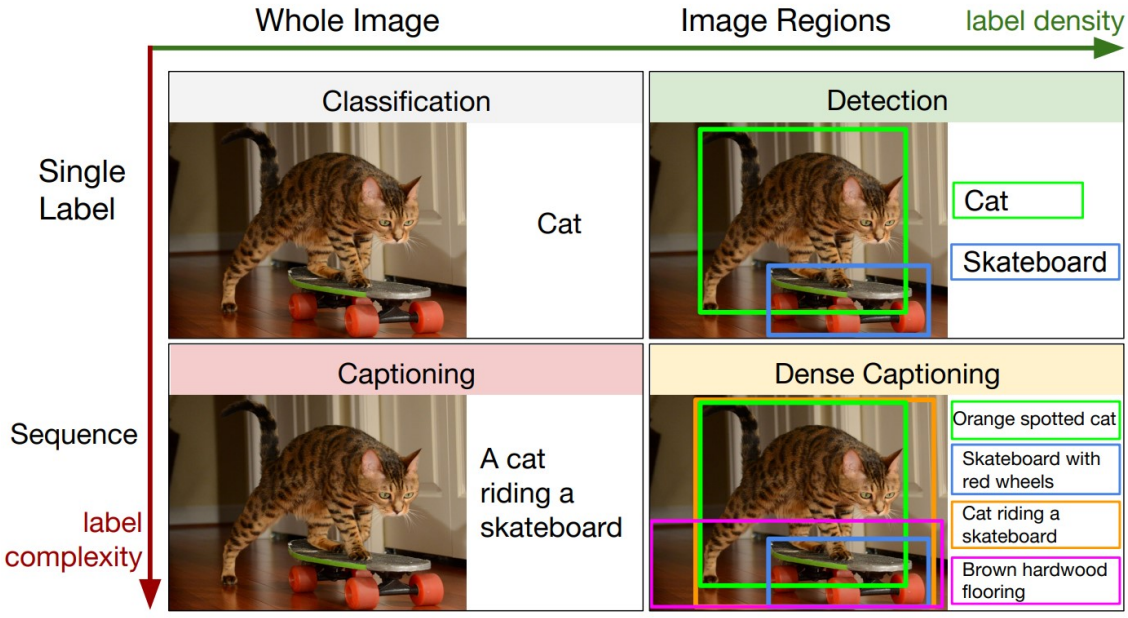


Figure 2-16. A deep learning application for image captioning, integrating work from both CV and NLP [83].

Figure 2-16 presents a ground-breaking work from the group led by Feifei Li [83], who developed a fully convolutional localisation network capable of describing not only a single image but also localised information in this image with a sentence of natural language. The integration of deep learning work from both CV and NLP significantly promotes machine "understanding" of its surrounding world.

Overfitting is a challenge in pure data-driven deep learning research, which frequently occurs when only scarce data are available or with improper training data structures. Besides, the inferior interpretability of the normal DNN has caused confidence shrinkage in its predictions, which confers it a title of "black box". Consequently, theory-guided learning using a physics-informed neural network (PINN) was developed to prevent overfitting problems and alleviate the low interpretability issue by respecting prior expertise and guiding the learning process with governing equations in relevant theories. The most prevailing form of physics-informed deep learning is the physics-informed data assimilation regression with physics-informed regularisation method. In this method, one or several governing equations in the research area, normally nonlinear partial differential equations (PDE), are used and converted to one or several physics-informed regularisation term in the objective function, which is configured in the same way as parameter norm penalty $\alpha\Omega(\boldsymbol{\theta})$ in Eq. (2-6). In [84–87], PINNs were used to predict numerical solutions of governing equations in computational fluid dynamics (CFD). It was reported to outperform pure data-driven DNNs with less training data. Besides of guiding deep learning with known governing PDEs, Tartakovsky et al. [84] used PINN to predict the unknown state-dependent coefficient in a nonlinear diffusion equation. Yao et al. [88] developed an FEA (finite element analysis)-Net which embeds the governing equation used in finite element analysis into the DNN design and consists of an FEA convolution network and an inference network, to predict FE simulation results for some simplified problems. These studies have shown the superiority of PINN over pure data-driven DNNs in generalisation accuracy and demand of training data.

For deep neural networks, the architecture of CNNs and RNNs has had constant development in the past decade. After the first deep CNN, AlexNet, was proposed by Krizhevsky [70], several CNN-based network structures were developed and became well-known, including VGGNet [89], GoogLeNet [90], ResNet [91], DenseNet [92]. In terms of

RNN architectures, apart from LSTM and GRU mentioned in Section 2.3.3, convolutional LSTM network [93] and attention-based RNNs [94] were also developed.

2.4 Machine learning: reinforcement learning and applications

Reinforcement learning is a technology which learns the optimal control strategy through active trial-and-error interaction with the problem environment. A reward is delivered by the environment as feedback for each interaction, and the goal of reinforcement learning is to maximise the total rewards. Almost all RL problems can be framed as a Markov Decision Process (MDP), which is defined as [95]:

$$\boldsymbol{M} = \langle \boldsymbol{S}, \boldsymbol{\mathcal{A}}, \boldsymbol{R}, \boldsymbol{P}, \gamma \rangle \tag{2-13}$$

where $\boldsymbol{S}$ is a set of possible states, $\boldsymbol{\mathcal{A}}$ is a set of possible actions, $\boldsymbol{R}$ is the reward function, $\boldsymbol{P}$ is the transition probability function and γ is the discounting ratio ($\gamma \in [0,1]$).

With the terms introduced above, the RL process can be briefly illustrated with a loop shown in Figure 2-17: from the state s_t at time t, an action a_t is selected based on the current policy, which leads to the next state s_{t+1} and a reward r_t for this step.

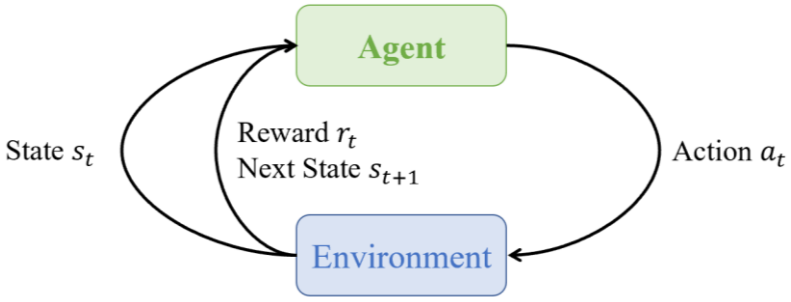


Figure 2-17. Reinforcement learning process.

To measure the goodness of a state, state-value and action-value (also called Q-value) are commonly used, which are respectively defined as follows [95]:

$$V^{\pi}(s_t) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t | s_t, \pi\right] \tag{2-14}$$

$$Q^{\pi}(s_t, a_t) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t | s_t, a_t, \pi\right] \tag{2-15}$$

where $\mathbb{E}$ denotes expectation and π denotes policy. The term $\sum_{t=0}^{\infty} \gamma^t r_t | s_t, \pi$ is the cumulative future rewards under policy π from *t*, known as return, of which the superscript and subscript denote exponent and time step, respectively. Thus, the optimal policy π^* is achieved when the value functions produce the maximum return, $V^*(s_t)$ and $Q^*(s_t, a_t)$.

Using Bellman's Equation [95], which decompose the value functions to immediate reward plus the discounted future rewards, the optimal value functions can be iteratively computed for every state to obtain the optimal policy, as shown below. This also indicates the condition of using these equations: the future only depends on the current state instead of history.

$$V^*(s_t) = \max_{a_t \in \mathcal{A}} \mathbb{E}_{s_{t+1}}[r(s_t, a_t) + \gamma V^*(s_{t+1})] \tag{2-16}$$

$$Q^*(s_t, a_t) = \mathbb{E}_{s_{t+1}}\left[r(s_t, a_t) + \gamma \max_{a_{t+1} \in \mathcal{A}} Q^*(s_{t+1}, a_{t+1})\right] \tag{2-17}$$

Two categories of RL algorithms have been investigated in this thesis, namely value-based and policy-based approaches. For the former, three Q-learning algorithms, namely deep Q-learning, double deep Q-learning and dueling deep Q-learning, are implemented. For the latter, three policy gradient algorithms, namely Advantage Actor-Critic (A2C), Deep Deterministic Policy Gradient (DDPG) and Proximal Policy Optimisation (PPO), are implemented.

2.4.1 Q-learning algorithms

As shown in Eq. (2-17), the optimal policy is obtained by iteratively updating the Q-value function for each state-action pair. However, it is computationally infeasible to compute them all when the entire state and action space becomes enormous. Thus, Q-learning algorithm [96] was brought up to estimate the Q-value function using a function approximator. Three function approximators were investigated in this thesis: Deep Q-Network (DQN), Double Deep Q-Network (Double-DQN) and Dueling Deep Q-Network (Dueling-DQN).

The objective function of DQN is defined as [96]:

$$J(\theta) = \mathbb{E}_{(s_t, a_t, r_t, s_{t+1}) \sim U(D)}\left[\left(r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \theta^-) - Q(s_t, a_t; \theta)\right)^2\right] \tag{2-18}$$

where $U(D)$ is a uniform distribution over the experience replay memory *D*. $Q(s, a; \theta)$ and $Q(s, a; \theta^-)$ respectively denote online network and target network, θ and θ^- are the network parameters. The term, $r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \theta^-)$, is the target Q-value at time *t*.

The objective function of Double-DQN is defined as [97]:

$$J(\theta) = \mathbb{E}_{(s_t, a_t, r_t, s_{t+1}) \sim U(D)} \left[\left(r_t + \gamma Q \left(s_{t+1}, \underset{a_{t+1}}{\text{argmax}} Q(s_{t+1}, a_{t+1}; \theta); \theta^- \right) - Q(s_t, a_t; \theta) \right)^2 \right] \quad (2\text{-}19)$$

where the selection of a_{t+1} is from the online network instead of target network as in DQN, and this policy is then evaluated by the target network for target Q-value. By decomposing the max operation in the target Q-value into two operations of action selection and action evaluation, the Q-values overestimation problem for DQN is alleviated.

The objective function of Dueling-DQN is the same as Eq. (2-18), while the dueling network factorises the Q-value evaluation to the evaluation of the state value and advantage value for better learning generalisation, which is defined as [98]:

$$Q(s_t, a_t; \theta, \alpha, \beta) = V(s_t; \theta, \beta) + \left(A(s_t, a_t; \theta, \alpha) - \frac{1}{|A|} \sum_{a_{t+1} \in \mathcal{A}} A(s_t, a_{t+1}; \theta, \alpha) \right) \quad (2\text{-}20)$$

where $A(s, a)$ denotes the action advantage-value, which is arithmetically related to state-value and action-value by $Q(s, a) = V(s) + A(s, a)$. The advantage-value measures the goodness of an action at a certain state. θ denotes the shared parameters of the dueling network, and α and β denote the exclusive parameters for the evaluation of advantage-value and state-value, respectively.

2.4.2 Policy gradient algorithms

Unlike Q-learning which achieves optimal policy by learning the optimal value functions, policy gradient algorithms parameterise the policy with a model and directly learn the policy. The objective function of policy gradient algorithms is configured to be the expected total return as shown by Eq. (2-14) and Eq. (2-15), and the goal of the optimisation is to maximise the objective function. Through *gradient ascent*, the policy model which produces the highest return yields the optimal policy. Most of the policy gradient algorithms have the same theoretical foundation, *Policy Gradient Theorem*, which is defined as [95]:

$$\nabla J(\theta) = \mathbb{E}^{\pi_\theta} [\nabla_\theta \ln \pi(a_t | s_t; \theta)\, Q^\pi(s_t, a_t)] \quad (2\text{-}21)$$

where ∇J denotes the gradient of the objective function. π_θ and $\pi(a_t | s_t; \theta)$ denote the policy model parametrised by θ.

The three policy gradient algorithms investigated in this thesis all use an Actor-Critic method [95] for policy update, of which the critic model is used for value functions evaluation to assist the policy update and the actor model is used for policy evaluation which is updated in the direction suggested by the critic.

The A2C algorithm uses temporal difference as the advantage term to assist policy update, whose gradient of objective function at time step t for actor and critic are defined as follows [99]:

$$\nabla J_{actor}(\theta)_t = \nabla_\theta \ln \pi(a_t|s_t;\theta)\left(r_t + \gamma V(s_{t+1};w) - V(s_t;w)\right) \tag{2-22}$$

$$\nabla J_{critic}(\theta)_t = \nabla_w\left(r_t + \gamma V(s_{t+1};w) - V(s_t;w)\right)^2 \tag{2-23}$$

where θ and w denote the parameters of actor model and critic model, respectively. The term $r_t + \gamma V(s_{t+1};w) - V(s_t;w)$ is the temporal difference advantage estimate, which is selected in this thesis through a preliminary study comparing it with Monte Carlo (MC) advantage estimate in this thesis.

The DDPG algorithm is specially developed for problems with continuous action space, whose gradient of objective function for actor is defined as [100]:

$$\nabla J_{actor}(\theta) = \frac{1}{N}\sum_t \nabla_a Q(s_t, a_t; w)\,\nabla_\theta \mu(s_t;\theta) \tag{2-24}$$

where N is the minibatch size at training. $\mu(\mathrm{s};\theta)$ denotes the deterministic policy model, differing from the stochastic policy $\pi(\mathrm{a}|s;\theta)$. It is noted that, instead of using advantage value to assist policy update in A2C, the gradient of Q-value with respect to action is used. The gradient of objective function for critic of DDPG is the same as that for A2C shown by Eq. (2-23) but with Q-value.

The PPO algorithm is a simplified version of Trust Region Policy Optimisation (TRPO) [101] by using a clipped objective function, which is defined as [102]:

$$J_{actor}^{clip}(\theta) = \mathbb{E}_{t\in N}\left[\min\left(r_t(\theta)\hat{A}_t, \mathrm{clip}(r_t(\theta), 1-\epsilon', 1+\epsilon')\hat{A}_t\right)\right] \tag{2-25}$$

where N is the minibatch size at training. $r_t(\theta)$ denotes the probability ratio $r_t(\theta) = \pi(a_t|s_t;\theta)/\pi(a_t|s_t;\theta_{old})$. $\hat{A}_t$ denotes the Generalised Advantage Estimate (GAE), which applied an additional discounting ratio λ for future advantage [103]. The gradient of objective function for critic of PPO is the same as that for A2C shown by Eq. (2-23) but with target state-

value computed by $\hat{A}_t + V(s_t; w)$. The clip function clips the probability ratio to be between $1 - \epsilon'$ and $1 + \epsilon'$ to prevent from extremely large parameters updates, which could lead to learning instability.

2.4.3 Applications

The development of reinforcement learning was impeded by the memory complexity and computational complexity before the advent of deep learning. With the powerful capability of DNN in function approximation and representation learning, these problems can be overcome and the research has been focusing on deep reinforcement learning (DRL) in the past decade [104].

DRL has two main applications: games and robotics. Benefit from the low-cost data acquisition, games are important testbeds for DRL algorithms, such as video games and board games. For video games, Mnih et al. [96] developed a learning agent, with deep Q-network, to learn policies directly from pixel inputs of classic Atari 2600 games, which was demonstrated to outperform all previous algorithms and achieve professional human-level control. Figure 2-18 presents the t-SNE embedding for the results from the last hidden layer of the learning agent, which describes how the learning agent recognise and "understand" the Atari game. The embedding is coloured by state values, which describe how good the state is in terms of scoring in the game. It can be seen that the agent has learned that full screen has the potential of getting more scores than partially completed screens. Inspired by the huge breakthrough, successful applications of DRL were also seen to Visual Doom [105] and StarCraft [106]. For board games, the most prominent application is AlphaGo, which was developed with deep supervised learning, DRL and Monte Carlo tree search (MCTS) [107]. Despite the gigantic search space and the hardness of position evaluation for Go, AlphaGo defeated the world champion Go player, Lee Sedol, 4 games to 1. Soon after, AlphaGo Zero was developed without any human knowledge for learning guidance, which achieved superhuman performance and defeated AlphaGo 100-0 [108]. Apart from Go, DRL was also successfully applied to Poker games [109,110]. The DRL applications to robotics focus on robotic manipulation and robot navigation, of which exemplary research include stacking blocks [111], five-fingered in-hand manipulation [112], visual navigation [113] and navigation through natural language instruction [114].

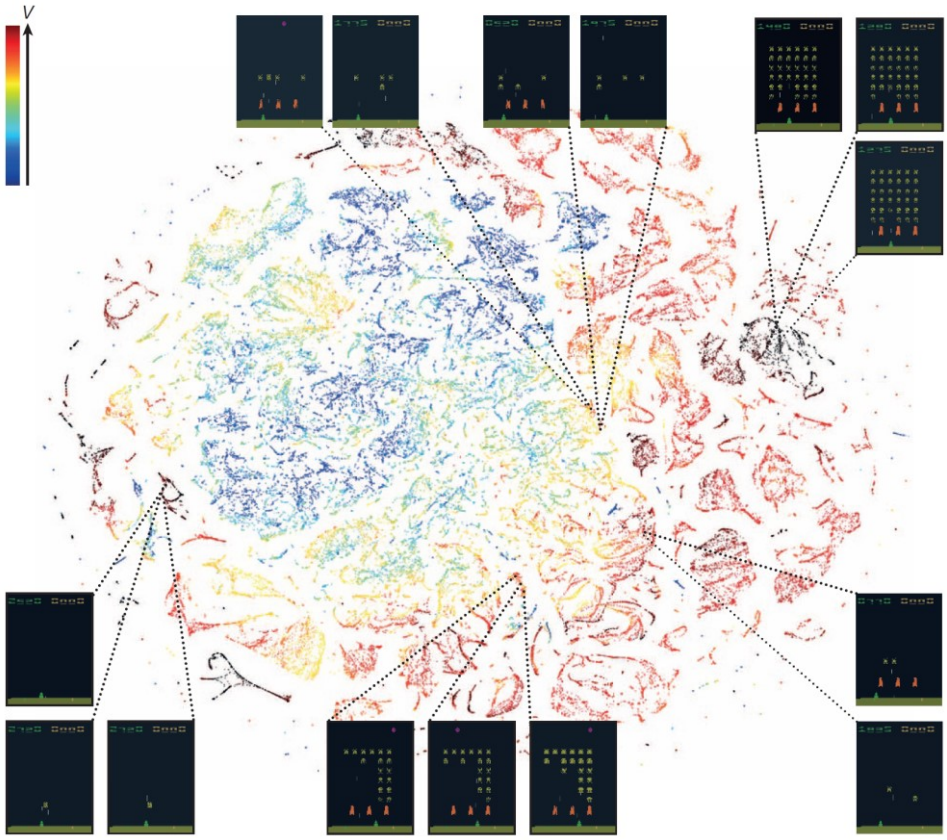


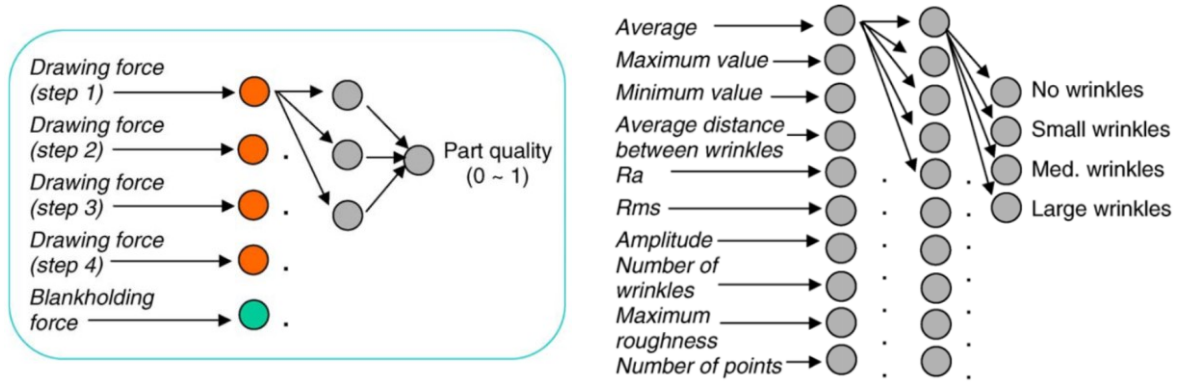
Figure 2-18. The t-SNE visualisation of the representations extracted from the last hidden layer of the learning agent, coloured by the state value [96] ***(black colour indicates the highest state value).***

2.5 Machine learning applications to sheet metal stamping process

As of today, the most prevailing machine learning applications to sheet metal stamping include process monitoring, fault diagnosis and surrogate assisted optimisation. In terms of process monitoring and fault diagnosis, machine learning models are developed to predict the process condition or manufacturing defects based on process-relevant parameters. Due to the forming characteristics of the traditional sheet metal stamping process, the sheet metal part is normally formed within a few or just one forming step, for which no research on tool path for stamping can be found. Most studies involving tool path generation and optimisation in sheet metal forming industry were performed for ISF process, which deforms the sheet metal to its target shape with a sequence of incremental deformations. Since deep neural networks are the primary learning models used in this thesis, only network-based machine learning approaches are reviewed in this section.

2.5.1 Process monitoring and fault diagnosis

The stamping parts quality, such as mechanical properties and occurrence of defects, is of great relevance to processing parameters. To improve the reliability and stability of sheet metal stamping process, machine learning techniques were utilised to monitor the stamping process by analysing process parameters and predicting forming results in real time. The applications can be classified into three groups based on the monitoring objects: formed parts, forming tools and forming process.



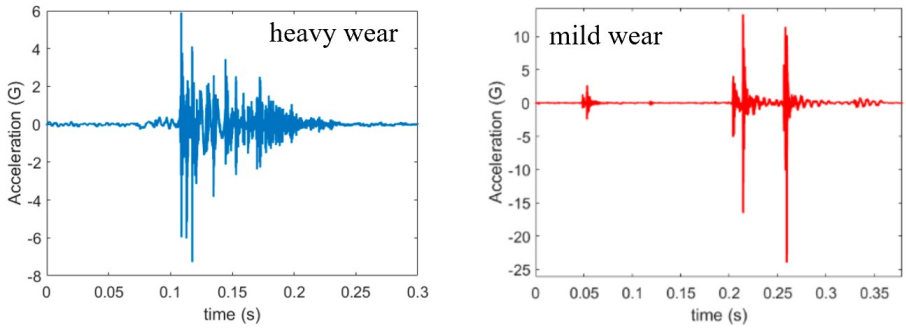
(a) MLP for cracks detection **(b) MLP for wrinkle classification**

Figure 2-19. Neural networks for process monitoring of a deep drawing process [115]**.**

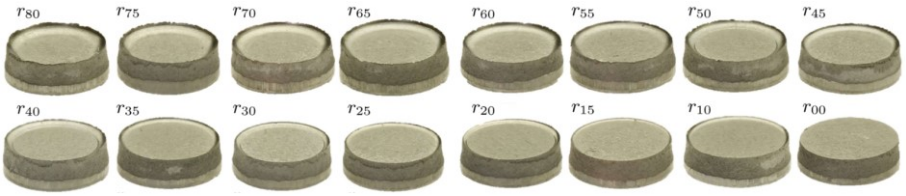
An early application monitoring formed parts was made by García [115], who integrated neural networks in a monitoring system to detect the occurrence and identify the category of defects in a deep drawing process. The real-time feedback from the neural network was transmitted to a control system for automatic modification of blankholding force and force distribution, through which high-quality parts can be constantly produced. Two MLPs, with one hidden layer, were implemented for crack detection and wrinkle classification as shown in Figure 2-19, of which the crack was predicted from drawing forces and blankholding force (process parameters) while wrinkle classification was predicted from handcrafted features of wrinkle profiles after drawing (process results). The neural network embedded automatic control system was reported to work properly at industrial requirements, with the detection accuracy of winkles and cracks being 92% and 89%, respectively. In addition to formed part failure, Zhao and Wang [116] utilised a three-layer MLP to monitor the material properties of formed parts through process parameters in a deep drawing process. Dib et al. [117] compared the performance of MLP, random forest, decision tree, naive Bayes, SVM, KNN and logistic regression in predicting springback and maximum thinning in a U-channel and square cup

forming processes from material properties and process parameters. Among these models which were trained with the same dataset, MLP was reported to be the best in identifying the springback, with slightly higher score than SVM.

Instead of using process parameters, Tsuruya et al. [118] inspected the product quality through acoustic emission (AE) signals. They attached an AE sensor to a support plate of the punch and collected AE signals through a deep drawing process. The count rate of the AE signal was fed into a four-layer MLPs to recognise the occurrence of crack of the formed part. The results showed that this approach achieved the recognition rate of 97.3%. More recently, deep neural networks were also used for defect detection. Inspired by work of object detection in computer vision, Block et al. [119] used a ResNet-based U-Net to detect imprint defects on stamped parts surfaces and classify their severeness. The network inspected defects through a high-resolution image of the formed part, and defect detection accuracy and severe imprint classification accuracy attained 76% and 90%, respectively.



(a) Vibration signal during stamping [120]



(b) Formed part images after blanking [121]

Figure 2-20. Different bases for tool wear monitoring in a stamping process.

There are also numerous machine learning applications to defect detection of forming tools. Huang and Dzulfikri [120] developed a one-dimensional CNN to monitor the tool health during the real-time stamping process. In their work, the vibration signals during stamping, as shown in Figure 2-20a, were measured for training the CNN to predict the class of the tool wear condition. The vibration signals were transformed to frequency spectrum with normalised

amplitude using Fast Fourier Transform before input to the CNN. The results revealed that the tool wear classification accuracy achieved up to 99.8%. Similarly, Griffin et al. [122] utilised unsupervised and supervised learning techniques to classify galling wear on sheet metal stamping tools, which used the AE features through the stamping process as input. The classification accuracy of the unsupervised and supervised learning techniques was reported to be 68% and 96%, respectively. In addition to process signals, formed part image was also used as the basis for tool wear monitoring. Molitor et al. attempted 27 state-of-the-art CNNs [123] and 3 generative adversarial networks (GANs) [121] in classifying 16 different wear states of the punch in a blanking process from the blanked workpiece images, as shown in Figure 2-20b. In the applications, the deep CNNs and GANs both achieved the classification accuracies of up to 99%.

For forming tools monitoring, apart from tool wear detection, tool life estimation was also attempted through machine learning approaches. For example, Kashid and Kumar [124] developed a three-layer MLP to predict the life of punches in a piercing and blanking process based on the maximum and minimum principal stress in the punch, of which the tool life is measured in terms of number of punching cycles. The training data were collected through FE computation, and the MLP prediction results were compared with analytical results, which showed a maximum deviation of 7%. Chen et al. [125] utilised an MLP with 3 hidden layers for life period estimation of a stamping press with punch sound recorded through the stamping process, which achieved a maximum prediction accuracy of 92%.

Instead of monitoring formed part and forming tools, a few applications were found detecting forming process anomalies. Lejon et al. [126] evaluated the performance of three machine learning techniques, namely autoencoder, SVM and isolation forest (IF), in anomaly detection in a hot stamping process, of which the autoencoder was reported to be the best and achieving 100% detection accuracy. The training data included the history of punch speed and punch force during stamping, and the autoencoder was trained to reconstruct the input data through an encoder-decoder architecture. A threshold was determined based on the reconstruction error from training, and any input signal having a reconstruction error exceeding this threshold was classified as anomaly. Similar to utilising autoencoder, which “calculates” the maximum similarity distance between normal process signals, Zhang et al. [127] proposed

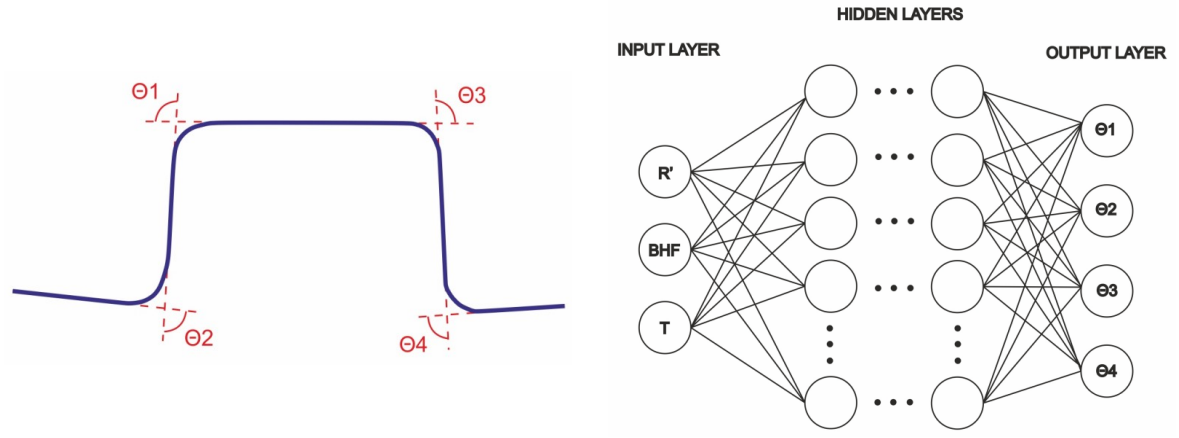
a semi-supervised clustering technique for anomaly detection in a punching process, using the strain signal on press column surface.

2.5.2 Surrogate assisted optimisation

A surrogate model is used in engineering when an outcome of interest is not well understood and cannot be measured or computed from its causes. Since machine learning is well-known for its capability of learning the intrinsic function between causes and outcome, it is most widely used to develop a surrogate model in sheet metal stamping industry. The review in this section only focuses on stamping process, surrogate modelling for material characterisation is not included. Based on the modelling target, the machine learning applications can be classified into two groups: process effect (springback) and FE results.

Springback is a common phenomenon in sheet metal stamping processes, for which vast research has attempted to develop a machine learning based surrogate model for fast springback prediction. The surrogate model predicts the springback for given process parameters, which provides a novel approach for process optimisation and springback compensation. An early application was made by Inamdar et al. [128], of which the springback angle of an air bent sheet metal part was predicted by a three-layer MLP using the angle of bend, tool dimensions and material properties. The model was trained with experimental data, which were collected from bending experiments with two punches, three dies and five different materials. The test evaluation revealed that the model had a maximum prediction error of 14%. In a later research, punch travel and punch velocity were also used as inputs to the surrogate model [129]. Liu et al. [130] proposed a novel technique, named GA-ANN, for springback prediction of U-shaped bent parts, which optimise the ANN parameters through genetic algorithm (GA). The GA-ANN was reported to achieve the prediction accuracy of 96%, which outperformed a previously developed regression model. Another novel technique, named BPNN-Spline, was proposed by Guo et al. [131], which combined an error back propagation neural network (BPNN) and spline function to predict the springback angle in a V-die bending process. The BPNN-Spline model was demonstrated to outperform traditional ANN. More recently, Spathopoulos and Stavroulakis [132] attempted surrogate modelling of an S-Rail forming process as shown in Figure 2-21, which had more complicated formed shape than previous applications. The model

was trained with FE data, and the evaluation results showed that the springback prediction accuracy was not consistent with different data sample, ranging from 60% to 99%.



(a) S-Rail formed part **(b) MLP predicting four springback angles**

Figure 2-21. The formed part through S-Rail forming process and the surrogate model for its springback angles prediction [132].

Machine learning based surrogate model is also widely used to replace FE computation and realise fast optimisation cycle. In traditional production cycle, the feasibility of a new process design is normally first verified through FE computation, followed by further experimental verification. However, FE computations for complex process and parts can be very time-consumptive, which necessasitates the development of surrogate model to shorten production cycle. As of today, there are two dominant approaches to construct surrogate models using machine learning methods: scalar-based method (SBM) and image-based method (IBM). SBM downscales either or both of the design variables (model input) and computation results (model output) to scalar-based metric, while IBM adopts image-based metric to retain the full spatial information. For SBM, an early surrogate model was developed by Singh and Kumar [133] using a three-layer MLP. The MLP predicted the thickness strains in three directions along the cup wall in a deep drawing process for forming limit estimation, using workpiece/tool geometries, material properties and process parameters. The model was trained with experimental data and evaluated by comparing its prediction with both experimental and FE simulation results. The comparison revealed that the results from the surrogate model were closer to experiments than those from FE simulation, which demonstrated the significance of using surrogate model as an alternative to FE for fast optimisation. Similar surrogate models for forming limit estimation were also constructed through strain prediction [134] and maximum thinning and thickenning rates predictions [135]. Hart-Rawung et al. [136] exploited

a deep MLP with 50 hidden layers as the surrogate model to replace the traditional austenite decomposition model for faster hot stamping simulations. The results showed that the surrogate model managed to predict the final phase fraction within about 85% accuracy on average.

Surrogate modelling with IBM was proposed to improve learning generalisation by providing exhasutic spatio information, which was inspired by the advance of deep CNN. Attar et al. [137] attempted a variant of U-Net, namely Res-SE-U-Net, as the surrogate model for a hot stamping proecss, as shown in Figure 2-22. The die geometry, blank geometry and stamping parameters were pre-processed to images and concatenated as the input to the model, which generated the thinning and displacement field of the formed part. Instead of only predicting the maximum thinning in SBM applications, the thinning distribution on the formed part was generated. This study revealed that not only did the image-based surrogate model had higher prediction accuracy than scalar-based one, it was also more favorable for its ability to construct formed part geometry with local thinning field. A similar study was also performed by Zhou et al. [138], who compared the performance of IBM and SBM in predicting plastic strain field using Res-SE-U-Net and eight-layer MLP, respectively. It was reported that the image-based U-Net outperformed the scalar-based MLP in prediction accuracy, generalisability, robustness and informativeness.

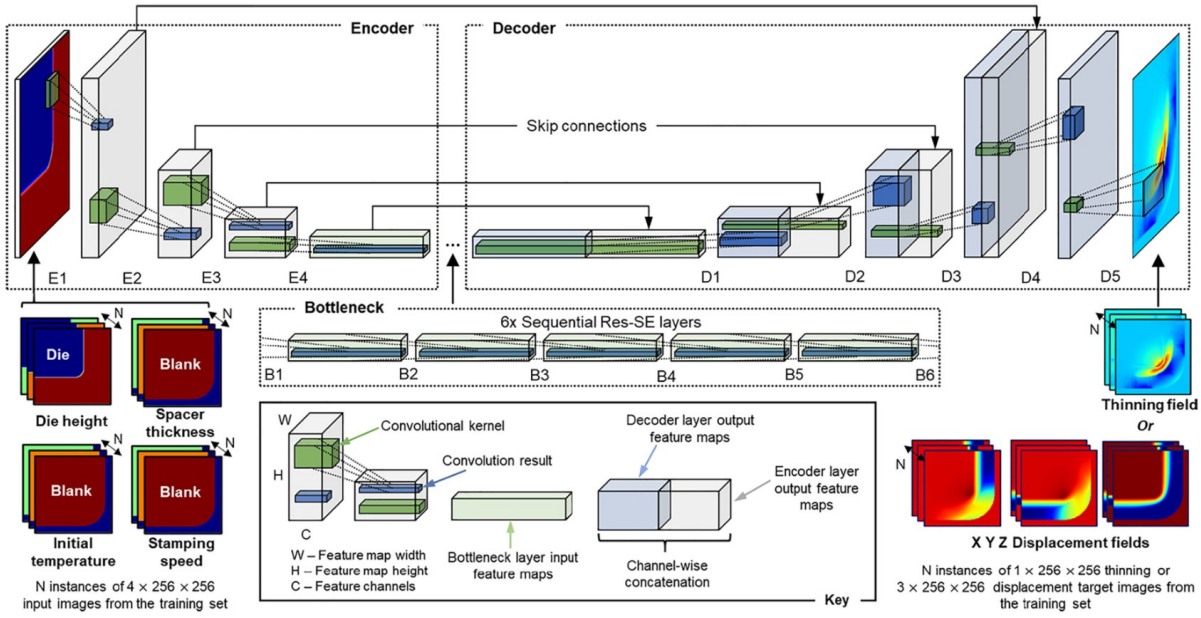


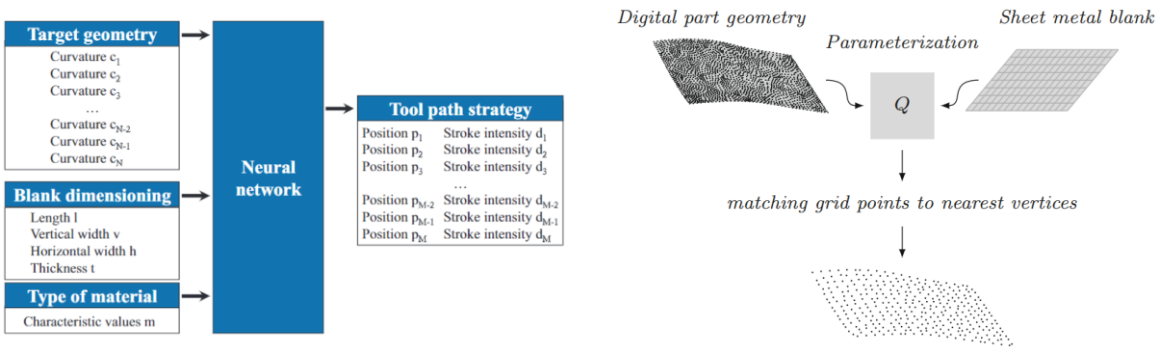
Figure 2-22. Res-SE-U-Net: an image-based surrogate model for sheet metal stamping [137].

The surrogate models introduced above were all constructed for forward engineering problems, i.e., from causes to outcomes. On the contrary, surrogate modelling for inverse

engineering problems was also found. For example, Roy [139] used a three-layer MLP to predict the forces required for a sheet metal bending process to achieve a zero springback angle, which achieved about 85% generalisation accuracy. Ivezic et al. [140] used an MLP to predict the die geometry required for a user-specified part shape. More recently, a deep learning approach was also used to identify the forming process for a given formed part geometry [141].

2.5.3 Tool path planning

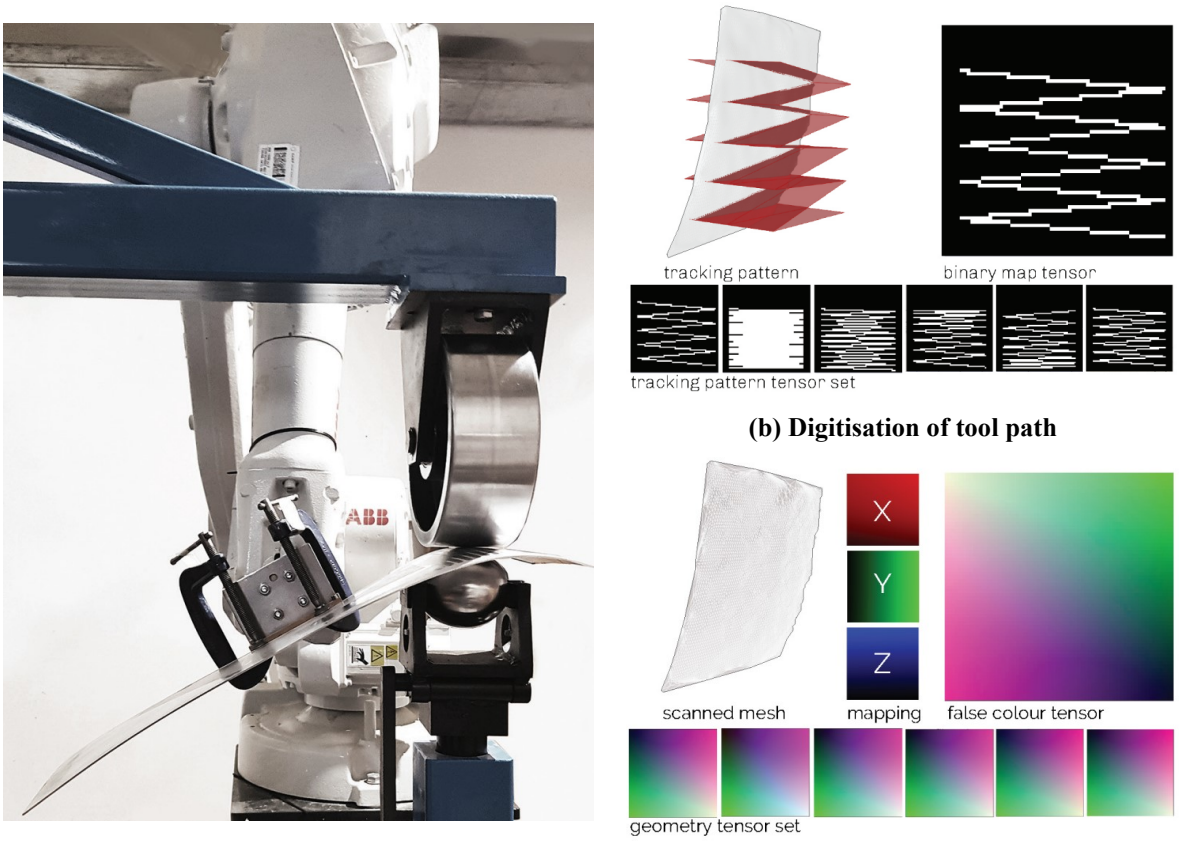
As introduced at the beginning of Section 2.5, machine learning applications of tool path planning in sheet metal forming industry are only found to flexible incremental free forming. As there is no research found using machine learning techniques to tool path planning for automated hammering processes introduced in Section 2.2, applications to other flexible forming are reviewed in this section. Opritescu and Volk [142] used an MLP to predict the tool path strategy for a 2-D automated driving process (Kraftforming), as shown in Figure 2-23a. The target workpiece geometry was discretised with equidistant positions, at which the curvatures were computed. The curvature distribution of the target geometry was fed into the network with the blank dimension and material. The blank was also discretised to have a fixed feed rate and tool movement path, on which the stroke intensities were predicted by the network. The model evaluation showed that the tool path prediction achieved an accuracy of over 99% (1 mm dimension error for workpiece with length of 350 mm). Hartmann et al. [143] extended this work to predict the tool path strategy for 3-D automated driving, of which the curvature for each mesh node was computed from the first and second fundamental form of local surface. To determine the tool movement path on the sheet metal blank, a conformal parameterisation method was used to map the formed part surface and blank surface to a unit square, as shown in Figure 2-23b. The evaluation results revealed that the tool path prediction achieved a similar level to the 2-D driving application.



(a) Input/output data for tool path prediction [142] **(b) Parameterisation of target workpiece** [143]

Figure 2-23. The neural network and data processing in an automated driving process.

Automated wheeling process with English Wheel was also attempted using machine learning techniques by Rossi and Nicholas [144], as shown in Figure 2-24a. The sheet metal was rested between the English Wheels and manipulated by a robotic arm to move back and forth in a zigzag pattern. The tool path was tracked and digitised to a binary map as shown in Figure 2-24b, and the formed workpiece was digitised by mapping the distribution of point coordinates to an RGB image as shown in Figure 2-24c. A fully convolutional network (FCN) was used to learn the intrinsic relationship between the tool path and its deformed part geometry. The model evaluation revealed that the test accuracy of predicting tool path given target workpiece geometry was 75% and that of predicting deformed part geometry given tool path was 65%.



(a) Automated wheeling

(b) Digitisation of tool path

(c) Digitisation of workpiece geometry

Figure 2-24. The experimental test setup and data processing in an automated wheeling process [144].

For ISF processes, tool path is mostly generated by contour generation algorithms, such as Stereolithography (STL) model, while machine learning applications to tool path planning are

limited. Störkle et al. [145] exploited a reinforcement learning algorithm (not specified by the author) to adjust the tool path and managed to increase the geometric accuracy of the product. However, they did not adjust the tool path directly but had the learning agent predicted a desired sheet metal geometry, the robotic system automatically generated the tool path for the predicted geometry. The sheet metal geometry was iteratively optimised within a learning loop as shown in Figure 2-25: the learning agent predicted the intermediate geometry based on the geometric deviation from the target at each iteration step, with which the intermediate tool path was generated for morphing the sheet metal until a prescribed geometric accuracy attained. They also attempted machine learning regression models for tool path optimisation for an ISF process [146], where the models included linear regressor, decision tree, random decision forest, SVM and Gaussian process regressor. Similar to their last application using RL, the tool path was generated automatically, while the local support force and support angle along the tool path were predicted by machine learning models, as shown in Figure 2-26, given tool path details and process parameters such as infeed depth and tool radii. Unfortunately, they did not report the geometric accuracy achieved using machine learning technologies.

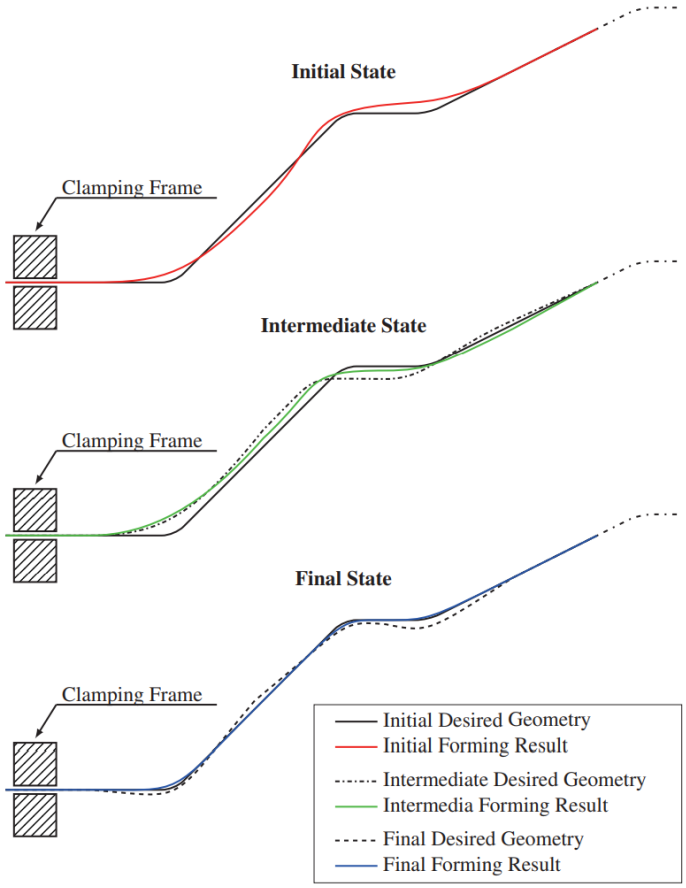
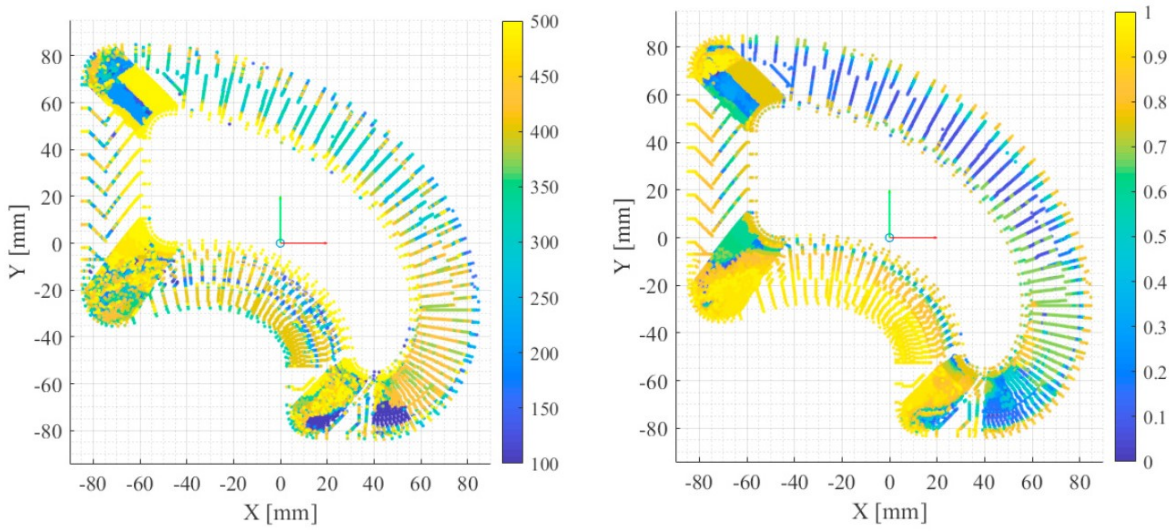


Figure 2-25. The sheet metal deformation progress through the ISF process [145].



(a) Support force (unit: N) **(b) Support angle (unit: %)**

Figure 2-26. Tool path with predicted (a) support force and (b) support angle for an ISF process [146].

2.6 Summary

Subject to the limited forming flexibility and high capital cost on customised punch and dies, the development of sheet metal stamping technology has been impeded. Thus, an emerging forming technique, namely automated hammering, has been brought up to replicate and automate the workflow of a blacksmith using bespoke machinery or industrial robotic arms. An exemplary automated hammering process is the ISP, of which the workpiece is clamped on an elastic support and struck by a centre hammer consecutively. To acquire more forming flexibility, novel fixtureless hammering processes with adjustable dies have also been developed.

Machine learning technologies can be classified into three categories: supervised learning, unsupervised learning and reinforcement learning. Supervised learning has exhibited its extraordinary learning and inference capabilities in computer vision and natural language processing fields, especially in classification and regression tasks. With the advent of deep learning, the deep neural networks have been found to be able to recognise high-level concepts and learn representations of data with multiple levels of abstraction. Using the state-of-the-art MLPs, CNNs and RNNs, tremendous advances in fields, such as image recognition, semantic segmentation, machine translation and speech processing and recognition etc., have been made. Yet, overfitting is a challenge in pure data-driven deep learning approach, especially when only scarce training data are available. Thus, theory-guided learning using a physics-informed neural

network (PINN) has been developed to respect prior expertise and guide the learning process, by introducing a physics-informed regularisation term in the objective function. Although PINN is rarely applied to metal forming industry, it has been reported to outperform pure data-driven DNNs with less training data in computational fluid dynamics (CFD) results prediction.

Reinforcement learning searches the optimal control strategy through active trial-and-error interaction with the problem environment. Using DNN as the function approximator, the obstacles of memory complexity and computational complexity in the development of RL are significantly alleviated. With DNN learning representations from inputs, the technique is then termed deep reinforcement learning. There are two main categories of algorithms in contemporary RL: value-based and policy-based approaches. The former searches the optimal policy by learning the optimal value functions, while the latter parameterises the policy with a model and directly learns the policy. DRL has been vastly applied to games and robotics field, where training data can be easily acquired. Well-known DRL applications include AlphaGo and AlphaStar from Google DeepMind.

Machine learning techniques have been applied to sheet metal stamping process for over two decades, of which the dominant applications include process monitoring, fault diagnosis and surrogate assisted optimisation. For process monitoring and fault diagnosis, most research focuses on monitoring the health of formed parts, forming tools and forming process. The bases for monitoring include process parameters, process signals and formed part images. Machine learning models have been used to predict the state of the monitored objects, and the performance of DNNs, in general, is found to outweigh that of shallow MLPs. For surrogate assisted optimisation, to shorten production cycle, a surrogate model has been constructed using neural networks for fast prediction of forming results of interest given the process parameters. Most studies are found to focus on surrogate modelling for springback and FE simulation results prediction, of which the former is mostly conducted using shallow MLPs while the latter is increasingly being researched using state-of-the-art DNNs. There are two dominant approaches to construct surrogate models for FE modelling, namely scalar-based method (SBM) and image-based method (IBM). Research has revealed that the IBM precedes SBM in prediction accuracy, generalisability, robustness and informativeness. In addition to surrogate modelling for forward engineering problems, applications to inverse problems are also found.

Since stamping processes normally deforms the sheet metal within a few or just one forming step, machine learning applications of tool path planning are only found to flexible incremental free forming processes. The flexible forming processes using machine learning include automated driving, automated wheeling and ISF. Supervised learning neural networks are commonly used for tool path planning, while reinforcement learning applications have been significantly ignored. This could be due to the expensive acquisition of computational or experimental data for RL algorithms training.

Chapter 3

Intelligent flexible sheet metal forming system architecture

3.1 Introduction

Inspired by the impressive learning capability of machine learning technology, it is exploited to explore the intelligence of sheet metal forming process. Figure 3-1 shows the concept of realising the intelligent sheet metal forming system through machine learning technologies. The inputs to this system include the target workpiece geometry the user expected to obtain and the material of the workpiece. The system is composed of three main parts, namely information, models and algorithms, of which the models learn the optimal forming tool path from the information in the way instructed by the algorithms. The information includes factual data, theoretical knowledge and heuristics from prior experience, of which the factual data provide the learning data basis while the knowledge and heuristics are supplied to facilitate the model learning process and improve the interpretability of the system intelligence. The models include state-of-the-art neural networks, which can have different architectures such as MLP, CNN, graphical neural network (GNN) and LSTM model. With regard to the algorithms, there are a number of selections for different learning purposes, which have been classified into essential and advanced ones. The output from this system is the forming tool path to achieve the target workpiece shape from a sheet blank, including forming parameters, tool choice and relevant forming results along the path. Besides the prediction phase, the output results are also evaluated based on the dimensional accuracy of the deformed workpiece through the predicted tool path. The evaluation feedback is then used to further improve the system, by feeding feedback results and new data to machine learning models and algorithms.

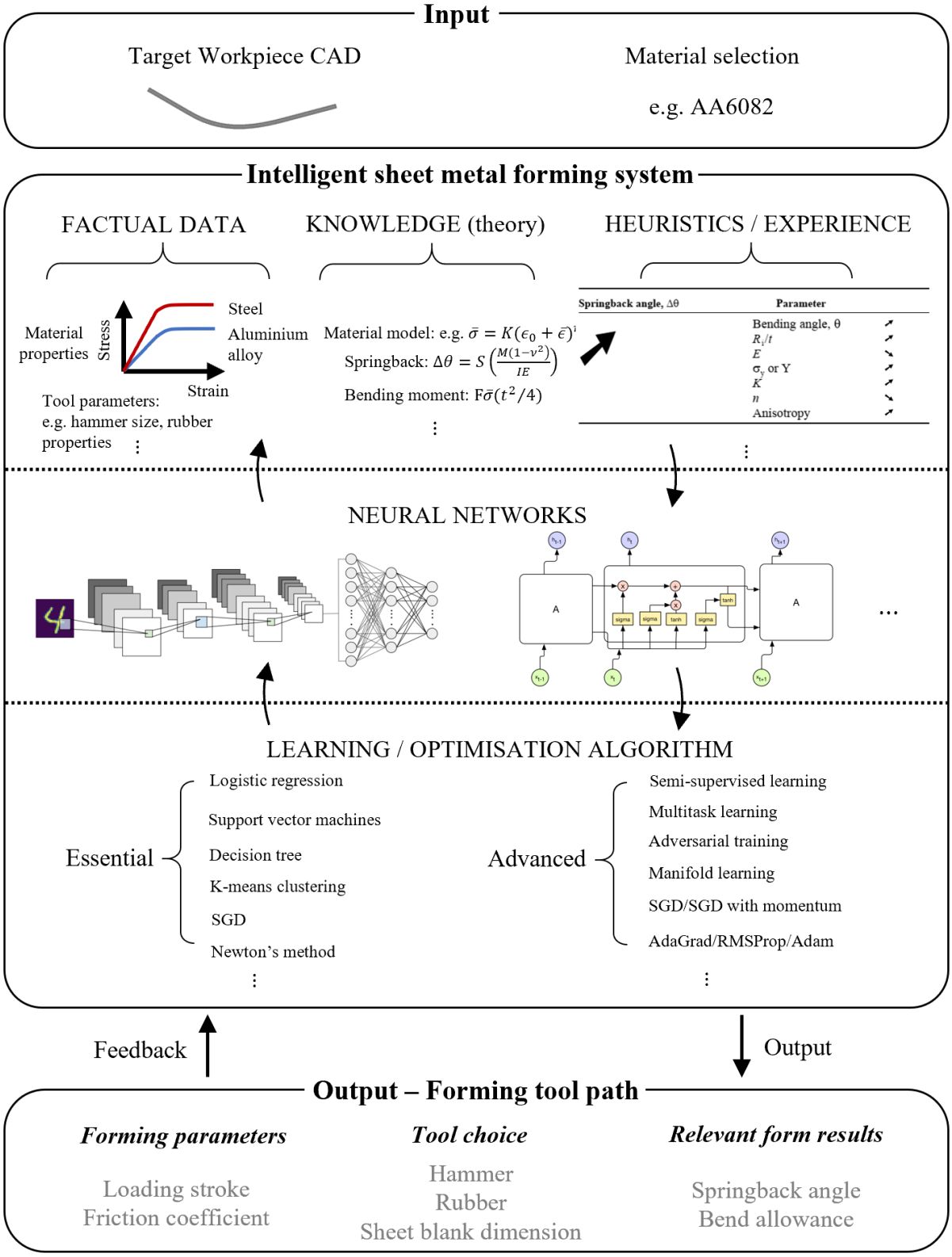


Figure 3-1. The concept of realising an intelligent sheet metal forming system through machine learning technologies.

3.2 The overall architecture

Based on the system concept illustrated in Figure 3-1, an intelligent sheet metal forming system is proposed to have the overall architecture shown in Figure 3-2.

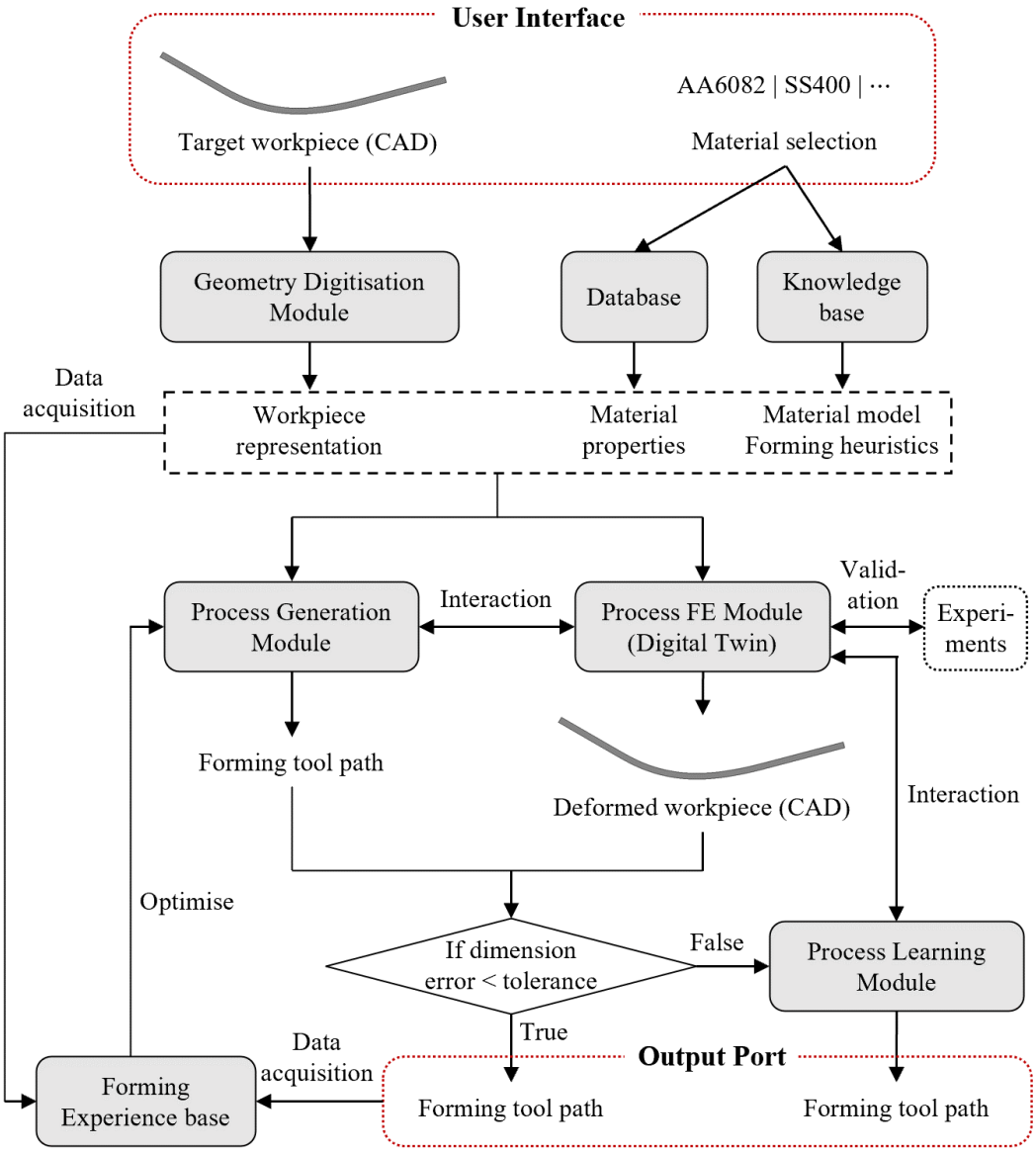


Figure 3-2. The overall architecture of the intelligent sheet metal forming system.

Human experts interact with the system through a user interface and an output port, where the target workpiece CAD as well as its material is input and the predicted forming tool path is obtained, respectively. There are seven modules designed for this system, namely Geometry Digitisation Module (GDM), Database (DB), Knowledge base (KB), Process Generation Module (PGM), Process FE Module (PFEM), Process Learning Module (PLM) and Forming

Experience base (FEB). After the inputs are fed into the system, the GDM digitises the target workpiece CAD to its representation form which is readily recognised by models. In accordance with the material selection, the material properties as well as relevant material model and forming heuristics are retrieved from the DB and KB, respectively. These data are then taken by the PGM and PFEM, where the forming tool path for the target workpiece along with the deformed workpiece geometry can be generated. The PFEM simulates the deformation process and computes the deformed workpiece geometry, which is taken by the PGM for tool path planning. The computational results from the PFEM are validated through experiments to ensure the applicability of the generated tool path in practical manufacture. The deformed workpiece is compared with its target shape, and the forming tool path is output through the output port unless the dimension error is greater than the prescribed tolerance. In the case of exceeding dimension error, the input data are fed into the PLM where the forming tool path for the target workpiece is learned with the assistance of PFEM. Once the satisfactory tool path has been generated from either the PGM or PLM, it is stored to the FEB along with the input data. The new data are then used to further optimise the models in the PGM for better tool path planning performance and generalisation.

The functionalities of the seven modules and their interactivities are listed as follow:

- Geometry Digitisation Module: it contains functions or algorithms converting the workpiece geometry of CAD format to its digital representation which is readily invoked by other modules.
- Database: it contains factual data relevant to the sheet metal forming process, such as material properties, forming tool/machine specifications.
- Knowledge base: it contains the theoretical knowledge and prior forming expertise and heuristics. The theoretical knowledge includes the theoretical functions or models in sheet metal forming processes proposed by researchers, such as material models and analytical equations for springback estimation. The forming expertise and heuristics include good forming operations and empirical rules learned during practical manufacture, such as factors affecting springback.
- Process Generation Module: it contains off-the-shelf models, which have been well-trained with current dataset, or empirical functions to generate the forming tool path for the target workpiece. In the process of tool path generation, the data of intermediate workpiece shape, which are computed through Process FE Module, could be needed.

- Process FE Module: it contains FE computation code or commercial software to simulate the sheet metal forming processes, which provides computational data for tool path planning and the deformed workpiece for dimensional accuracy evaluation. The accuracy of the FE computation is validated through experiments.
- Process Learning Module: it contains learning algorithms and models to learn the forming tool path for the target workpiece from scratch. It is only triggered active if the deformed workpiece through the tool path generated from the Process Generation Module cannot meet the dimensional accuracy requirement. The Process FE Module is used to provide the computational environment during the learning process.
- Forming Experience base: it stores all the forming tool path predictions and their prediction object in previous forming experience, which are used to optimise the tool path generation models in the Process Generation Module.

3.3 Deep learning applications to main modules

As illustrated in Figure 3-1, machine learning technologies are exploited to realise the intelligence of the sheet metal forming system. Among the seven modules in the system, the three main modules which directly contribute to tool path planning, namely Process Generation Module, Process Learning Module and Process FE Module, can be configured with machine learning techniques.

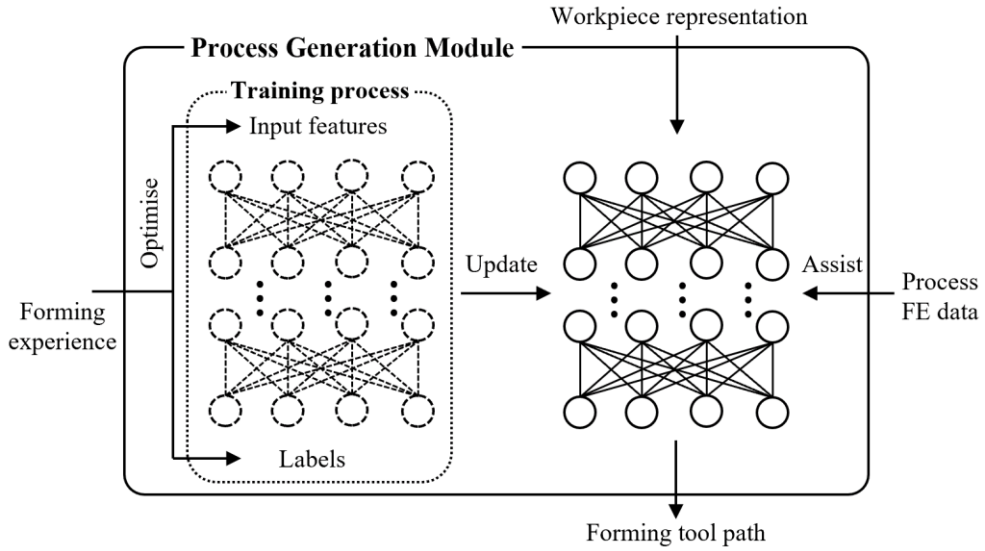


Figure 3-3. The architecture of the Process Generation Module.

Figure 3-3 presents a viable architecture of the Process Generation Module. There are two neural networks sharing the same architecture in the module, an online network and an offline network (in the dotted square). The online network predicts the forming tool path for the target workpiece input to the system by human experts, with the assistance of FE computational data. The offline network is trained with the previous forming experience data from the Forming Experience base through supervised learning, and it is further optimised once new experience is input. The model parameters of the online network will be updated by those of the offline network if the offline one has higher generalisation accuracy.

Figure 3-4 presents a viable architecture of the Process Learning Module, which exploits reinforcement learning techniques for tool path learning. In the RL process, the computation from the Process FE Module acts as the RL environment. The sheet blank geometry is input as the initial state of the RL process, which is taken by the learning agent to predict the forming actions (e.g., forming location and stroke). From the state and action, the RL environment computes the next state and evaluate the goodness of the state-action pair. The goodness is measured by a reward, which is determined by the contribution of the current state-action pair to the forming target. With the reward, the agent is updated to be able to distinguish the goodness of different state-action pairs. Through looping the RL process in the search space, the optimal tool path can be obtained.

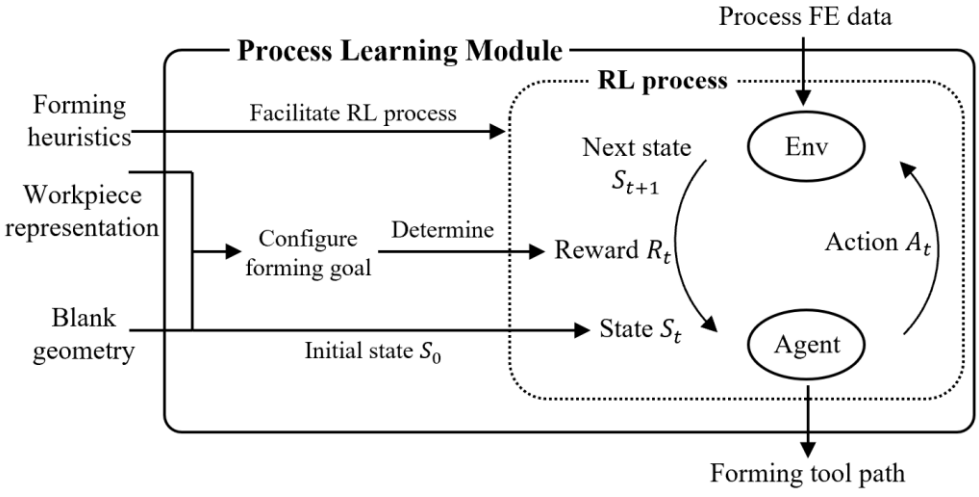


Figure 3-4. The architecture of the Process Learning Module.

Figure 3-5 presents a viable architecture of the Process FE Module, in which a surrogate model is used to replace the computationally expensive FE simulations. The surrogate model is an image reconstruction model with an encoder-decoder architecture, which takes all

deformation conditions and predicts the deformed workpiece representation. The predicted representation is then reconstructed to the deformed workpiece CAD file and output from the module. The prediction accuracy of the surrogate model is evaluated using FE simulation results, and the FE results are validated by experiments.

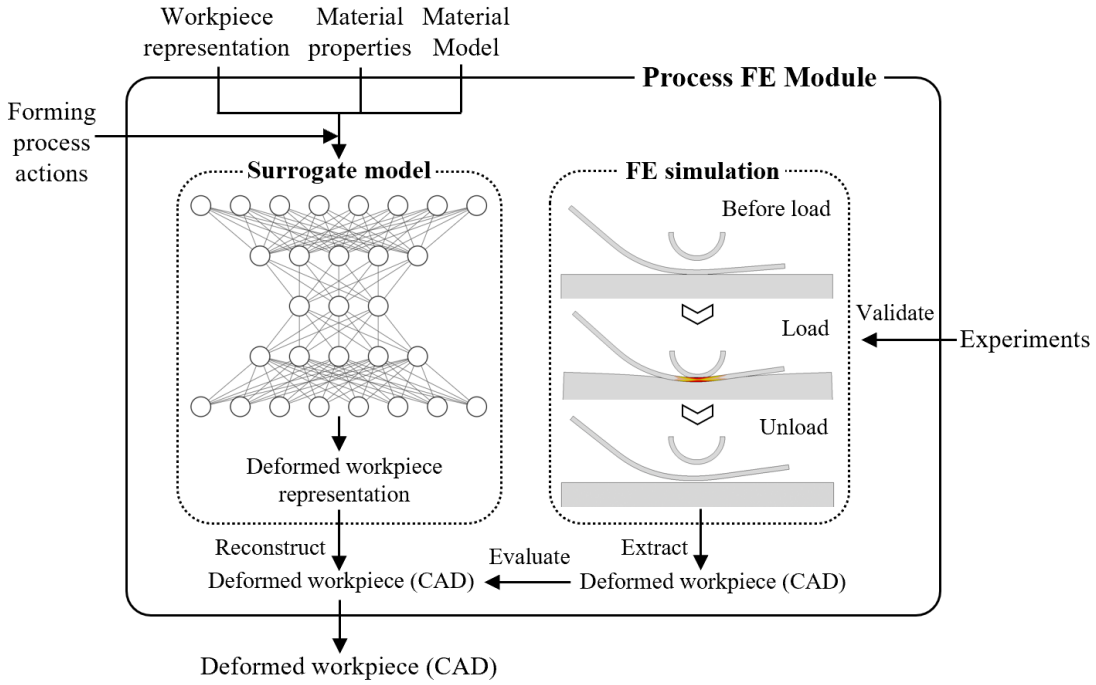


Figure 3-5. The architecture of the Process FE Module.

3.4 Summary

In this chapter, the concept of realising the intelligence of a sheet metal forming system by incorporating the state-of-the-art machine learning technologies has been introduced, followed by the overall architecture of the proposed intelligent system.

The system is designed based on the principle of modularity, so that each module of the system can be parallelly operated, controlled and developed. Seven modules in the system as well as their functionalities and interactivities are illustrated, covering the operations of data storage, raw data pre-processing, tool path planning, process computation and validation, post-processing and feedback mechanism. In addition, a dual modular redundancy (DMR) is designed for tool path planning with a Process Generation Module and a Process Learning Module to ensure the accuracy and efficiency of the intelligent system.

The viable machine learning embedded architectures of three main modules of the system, namely Process Generation Module, Process Learning Module and Process FE Module, are introduced. Deep supervised learning and reinforcement learning techniques are exploited to configure these modules, which contribute to the system intelligence.

Chapter 4

Single punch: theory-guided deep learning with scarce experimental data

4.1 Introduction

Subject to the high yield strength-to-elastic modulus ratio of high strength aluminium alloy and steels, the stamped sheet metal parts are predisposed to have notably large springback. To address this issue, conventional springback compensation methods have been attempted experimentally and numerically [40,147,148]. Machine learning based method has also been developed for the last two decades as an alternative approach, as introduced in Section 2.5.2. However, the development of pure data-driven machine learning model has been impeded due to its requirements of big data and low interpretability.

This chapter presents a novel theory-guided deep learning method, through multi-task learning and parameter norm regularisation, to alleviate the demand on the expensive experimental data and improve the model interpretability. This work has been presented at IEEE/CAA Journal of Automatica Sinica [149]. This work presents a series of numerical experiments to evaluate the efficacy of three machine learning models, namely support vector regressor (SVR), data-driven DNN and theory guided DNN (TG-DNN), for sheet metal bending. These models were trained to predict the required loading stroke for sheet metal bending processes, given only the CAD data of the desired workpiece to be produced. Section 4.2 presents the two bending processes investigated and their digitisation methods. Section 4.3 presents the rationale of the theory-guided learning method and the architecture of the DNN implemented. Section 4.4 presents the data acquisition process for model training, followed by

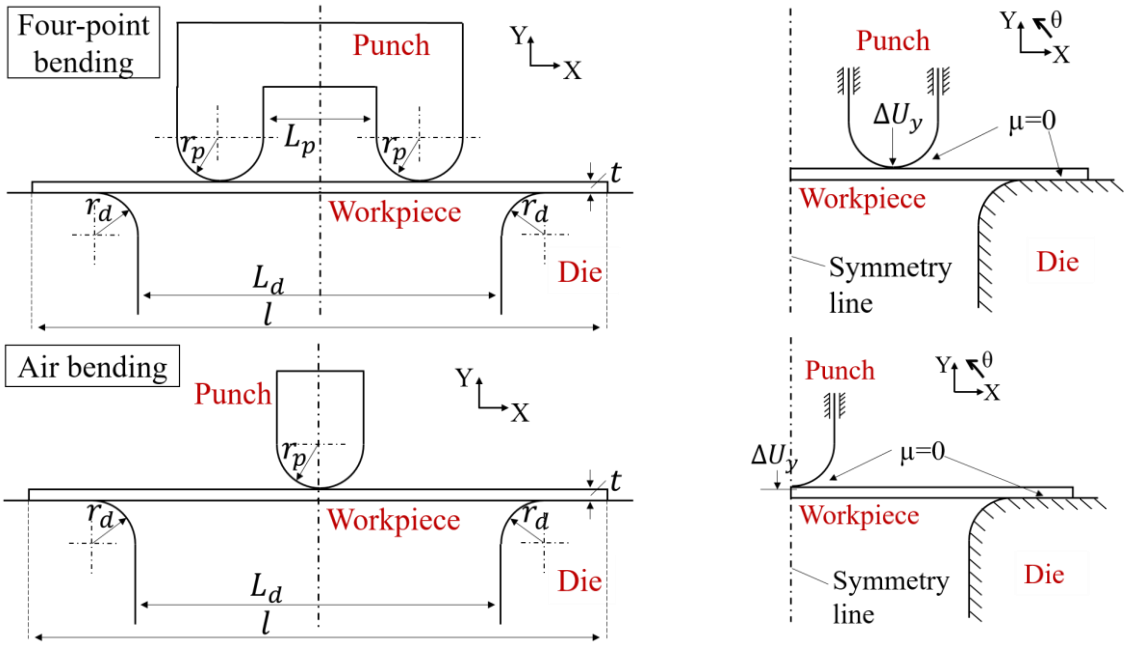
details of training methods. Finally, Section 4.5 presents the performances from the SVR, data-driven DNN and TG-DNN under various experimental conditions.

4.2 Experimental setup and forming digitisation

To comprehensively evaluate the performance of the data-driven DNNs and TG-DNN, two common sheet metal bending processes, four-point bending and air bending, were employed with two commonly used material for sheet metals, AA6082 [150] and SS400 [151]. Abaqus 2019 was used to perform the numerical experiments and export the experimental data.

4.2.1 Experimental setup

Figure 4-1 presents the FE models and experimental setups for four-point bending and air bending. As can be seen in the FE model of the four-point bending in Figure 4-1b, the punch was only allowed to translate along Y axis for a given displacement. The Die was fixed, and the workpiece was symmetric about its symmetry line. The friction coefficients between tools and workpiece in both cases were set to be zero for simplicity.



(a) The test setup **(b) FE model**

Figure 4-1. The geometry, test setup and plane strain FE model for the four-point bending and air bending.

The specification of the experimental setup for four-point bending and air bending in Figure 4-1a is presented in Table 4-1.

Table 4-1. Experimental setup data for four-point bending and air bending (unit: mm).

	Workpiece length, l	Workpiece thickness, t	Punch radius, r_p	Punch distance, L_p	Die radius, r_d	Die opening, L_d
Four-point bending	160	1	5	50	5	110
Air bending	160	1	5	-	5	110

As these numerical experiments were performed at 2-D scale, plane strain element with quadratic geometric order was used to mesh the part of workpiece. In Abaqus 2019, it is defined as CPE8R: An 8-node biquadratic plane strain quadrilateral, reduced integration. The punch and die working face were meshed with undeformable mesh, while the workpiece was set to be homogeneous and modelled with deformable mesh. A total element number of 21,328 was used in the FE simulation, after a grid-independence study performed to determine the most appropriate mesh size which assures reasonable simulation accuracy (over 95%) with low computational time.

The stress and strain curves of material of AA6082 and SS400 were extracted from [150,151], with which the main properties of these materials were summarised and listed in Table 4-2. The strain-hardening coefficient, n, and strength coefficient, K_s, were calculated by fitting the following equation to the data domain in log-scale stress-stain curve where plastic deformation occurs [152]:

$$\sigma = K_s \varepsilon^n \tag{4-1}$$

where σ is the true stress and ε is the true strain.

Table 4-2. Values of the parameters for the materials investigated.

Materials	Young's modulus, E (GPa)	Poisson's ratio, γ_p (-)	Density, ρ (g/cm^3)	strain-hardening coefficient, n (-)	strength coefficient, K_s (-)
AA6082	70	0.33	2.71	0.0731	400.22
SS400	210	0.26	7.86	0.1448	708.27

4.2.2 Forming digitisation

Figure 4-2 presents an example of the shape of a target component and its geometry representation for four-point bending and air bending, respectively. The top surface of the target component was extracted and split into a number of equidistant data points.

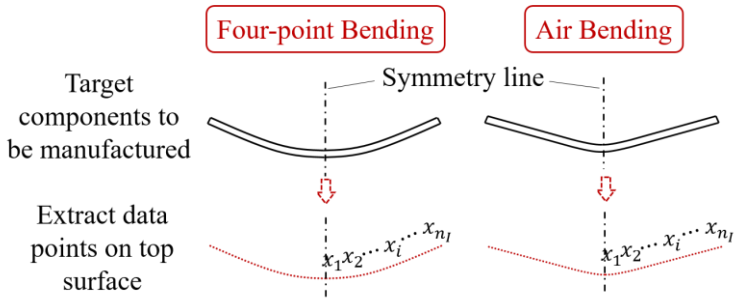
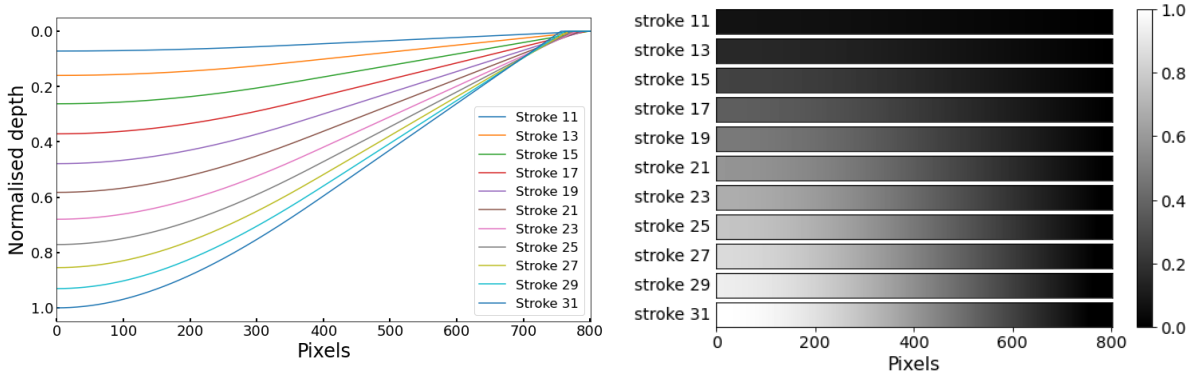


Figure 4-2. Examples of target components to be manufactured using four-point bending and air bending respectively and geometry representation for assessing the error between the designed and manufactured components.

Modern DNNs, especially the CNNs in Computer Vision, have been proven to be able to localise and extract the important features from image-based inputs, which have been confirmed to facilitate the learning of the relationship between input data and the outputs of interests [83,153]. With the hindsight, the input in the whole dataset was designed as an image-based representation, known as "grayscale map", with a 1-D array containing the local depth value about the free-end of the workpiece. Within the "grayscale map", grayscale-values (GS), ranging from 0 to 255, were used to encode the local depth. 0 GS ("black") denotes the location at and beyond the free-end of the workpiece (no depth), while 255 GS ("white") denotes the location with highest depth. To further improve the learning potential of DNNs, the grayscale maps were normalised by its highest value and the outputs were standardised in the output data domain.

Figure 4-3 presents some examples of target workpiece in their normalised shape and grayscale map. It is noted that the transverse dimension in grayscale maps is used only for better visualisation.



(a) Normalised depth vs longitudinal pixels **(b) Grayscale map**

Figure 4-3. Examples of workpiece geometry representations (801 pixels).

There were in total 801 pixels in the longitudinal dimension, evenly distributed and aggregated to be the length of 80 mm (half length of the original workpiece blank). The map values were interpolated, with inverse distance weighting (IDW) interpolation method, from the computational results of coordinates along half of the top surface of the workpiece after springback, leveraging the symmetry of the workpiece shape.

4.3 Theory-guided deep learning and neural networks

Theory-guided deep neural network respects prior treasurable theoretical or empirical knowledge by leveraging the theories to regularise the learning process. Compared to pure data-driven DNN model, a well-trained TG-DNN would be more than just a "*black box*" as theoretical knowledge is taken into account and the credibility of its prediction results is increased. Similar to the general form of the regularised objective function in Eq. (2-6), the theory-guided loss function is regularised by theoretical laws as follow:

$$\hat{J}_{TH}(\boldsymbol{\Theta}; \boldsymbol{X}, \boldsymbol{Y}) = J(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{Y}) + \alpha_{TH} J_{TH}(\boldsymbol{\theta}; \boldsymbol{X}) \tag{4-2}$$

where $\alpha_{TH} \in [0, \infty)$ is a hyperparameter used to control the relative contribution of the theory-guided penalty term, $J_{TH}(\boldsymbol{\theta}; \boldsymbol{X})$, which is defined as:

$$J_{TH}(\boldsymbol{\theta}; \boldsymbol{X}) = \frac{1}{n_{\mathcal{H}}} \sum_{i=1}^{n_{\mathcal{H}}} \left[\mathcal{H}(\boldsymbol{X}_i, \widehat{\boldsymbol{Y}}(\boldsymbol{X}_i; \boldsymbol{\theta}))\right]^2 \tag{4-3}$$

where $\mathcal{H}(\boldsymbol{X}_i, \widehat{\boldsymbol{Y}}(\boldsymbol{X}_i; \boldsymbol{\theta}))$ is a function taking in the inputs and outputs of the TG-DNN at one regularisation spot, which measures the divergence of the theoretical laws at this spot. $n_{\mathcal{H}}$ is the total number of regularisation spot. By minimising the loss between network outputs and labels as well as the divergence from the theoretical laws, the network would be guided to learn the underlying physics of the theories and generate outputs respecting these theories.

As stated in Section 2.3.4, the most common form of $\mathcal{H}(\boldsymbol{X}_i, \widehat{\boldsymbol{Y}}(\boldsymbol{X}_i; \boldsymbol{\theta}))$ for PINN, to date, is the governing equation in the research area, which is normally one or several nonlinear *partial differential equations* (PDE). The convergence of the regularisation term would guide the learning process by forcing the governing equation to hold between the inputs and outputs of the PINN. In this way, the PINN could learn the underlying theoretical laws and its prediction would be more interpretable than pure data-driven DNNs. However, due to the intrinsic complexity of Lagrangian mechanics in most practical engineering problems, the material

model, Swift's law, is used as the substitute for the PDE governing equations to regularise the network learning. By implementing this theory in training, the network could learn in a way respecting the material model. A multi-task learning network configuration was used to implement the regularisation. This method uses a less hard theory-guided penalty term $J_{TH}(\boldsymbol{\theta}; \boldsymbol{X})$ than the mainstream PINNs to regularise the learning of extra tasks to improve the learning of primary ones, instead of bridging the inputs and outputs with governing equations. Since the prediction of extra tasks (stress and strain) come from the same model parameter that predict the primary task (stroke), the training of model parameters is constrained by the theory. Multi-task learning has been employed in lots of research for its improved learning speed and accuracy and more chances for harder tasks to be learned than single-task learning [154], such as face detection and gender recognition in computer vision [155], natural language processing [156] and speech recognition [157]. The rationale behind is that the model with multi-task learning can learn related work at the same time and use these tasks as inductive bias for each other, which allows the domain regularities to be better learned [56].

Swift's law is a generalised power law for expressing the material-hardening behaviour when plastic deformation occurs in a material [5], which has been introduced in Eq. (4-1). To replicate material model obtained in the one-dimensional tension test, a plasticity model is defined using a scalar equivalent stress and an equivalent plastic strain to replace the true stress and true stress in Eq. (4-1). The von Mises stress, σ_v, and the corresponding equivalent plastic strain, ε_{eq}, are used in this study:

$$\sigma_v = K_s {\varepsilon_{eq}}^n \tag{4-4}$$

Thus, Eq. (4-4) is the theoretical equation used in this study to regularise the learning of the TG-DNN. Two extra tasks on learning additional physical properties (true stress and true strain) are added to the TG-DNN for regularisation. Consequently, Eq. (4-4) can be written into the form of $\mathcal{H}\left(\boldsymbol{X}_i, \widehat{\boldsymbol{Y}}(\boldsymbol{X}_i; \boldsymbol{\theta})\right)$:

$$\mathcal{H}\left(\widehat{\boldsymbol{Y}}(\boldsymbol{X}_i; \boldsymbol{\theta})\right) = \widehat{\sigma_v}(\boldsymbol{X}_i; \boldsymbol{\theta}) - K_s(\widehat{\varepsilon_{eq}}(\boldsymbol{X}_i; \boldsymbol{\theta}))^n \tag{4-5}$$

Then the theory-guided objective function regularised by Swift's law is given as follow:

$$\hat{J}_{TH}(\boldsymbol{\Theta}; \boldsymbol{X}, \boldsymbol{Y}) = J(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{Y}) + \frac{\alpha_{TH}}{n_{\mathcal{H}}} \sum_{i=1}^{n_{\mathcal{H}}} \left(\widehat{\sigma_v}(\boldsymbol{X}_i; \boldsymbol{\theta}) - K_s \left|\widehat{\varepsilon_{eq}}(\boldsymbol{X}_i; \boldsymbol{\theta})\right|^n\right)^2 \tag{4-6}$$

where an absolute value of the term $\widehat{\varepsilon_{eq}}(\boldsymbol{X}_i;\boldsymbol{\theta})$ is used to prevent the base of the exponent from being negative values, as n is less than 1. α_{TH} is the hyperparameter and $n_{\mathcal{H}}$ is the total number of regularisation spot.

The effectiveness of the introduced theory-guided penalty term, $J_{TH}(\boldsymbol{\theta};\boldsymbol{X})$, is manually checked for the training examples by comparing the calculated values of the penalty term and the ideal value (zero). The calculation check was performed for the whole 201 data in the four experiments shown in Section 4.5.2, with the relative error to the ideal value presented in Figure 4-4. It can be seen that the relative error for each training dataset is negligibly small, thus, the application of the theory-guided penalty to regularise the learning of the training data domain is theoretically valid.

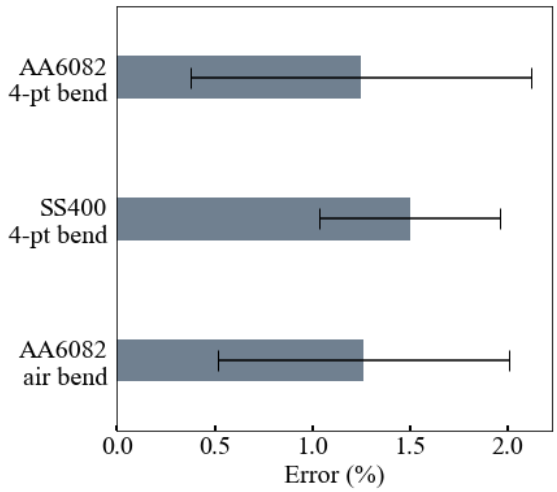


Figure 4-4. The relative error between the calculated results of the theory-guided penalty term for the whole 201 examples and the ideal value (zero) for the indicated materials and processes. 4-pt bend denotes four-point bending process.

Figure 4-5 shows the data flow of theory-guided learning in the intelligent sheet metal forming system shown in Figure 3-2. The database provides factual data to an FEA module where numerical computations are conducted to acquire the data for training of the TG-DNN. The learning of the DNN is guided by the prior knowledge or theories, which are stored in the knowledge base. After the DNN is trained to convergence, a new target workpiece geometry which is different from those in the training data is fed into the DNN through a user interface. A predicted stroke for the newly given geometry is then obtained from the DNN and used for another process simulation in the FEA module to yield the workpiece geometry after springback. Finally, the data points representing the predicted and target workpiece geometry are compared to determine the maximum distance between them, which is denoted by $\Delta GEOM_{pred}$.

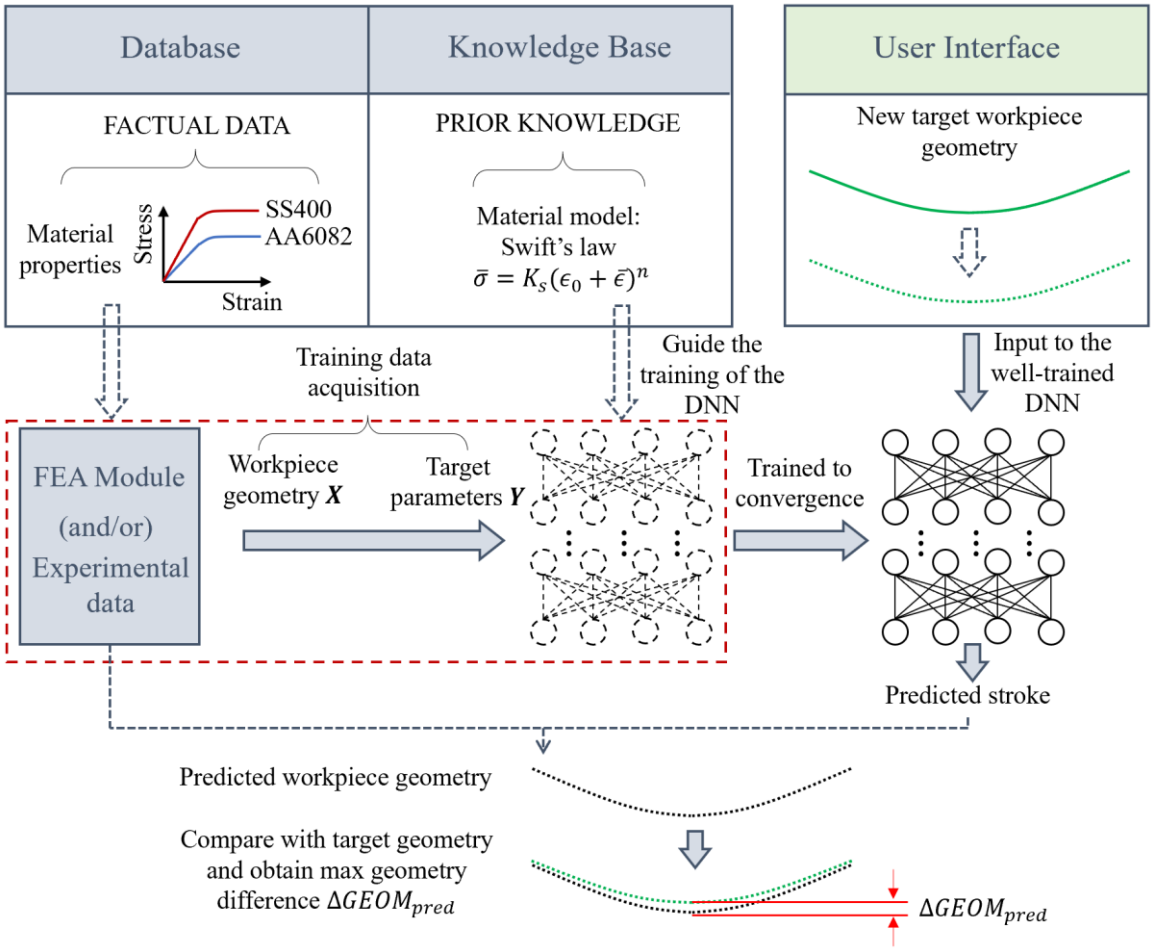


Figure 4-5. Theory-guided learning data flow.

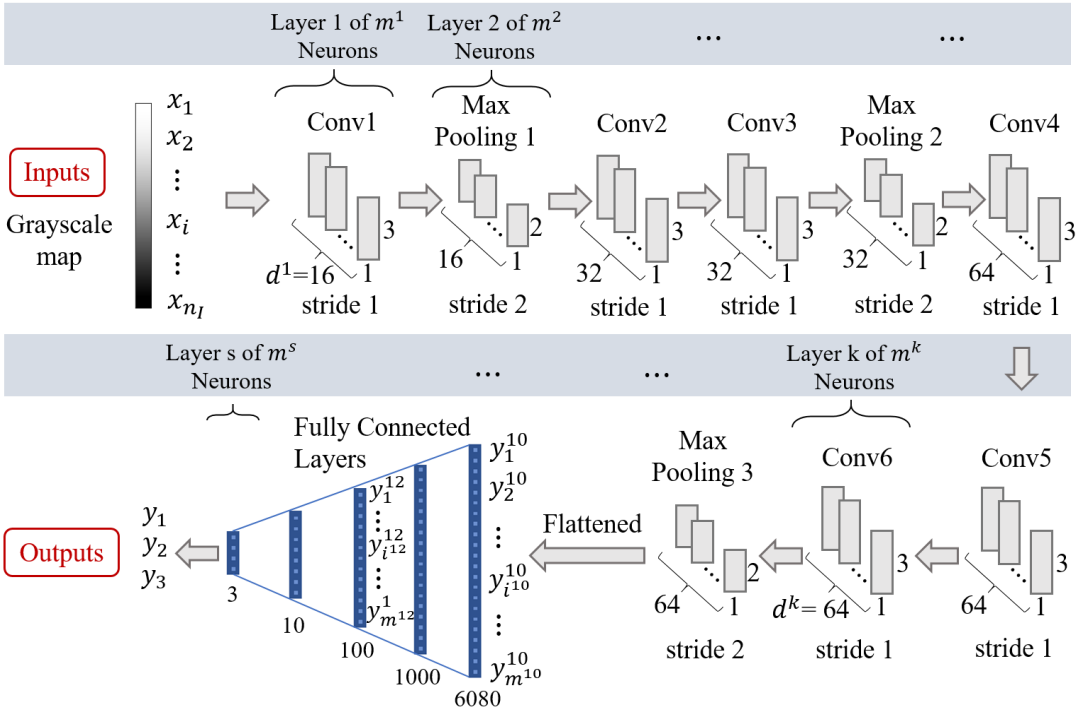


Figure 4-6. Architecture of the deep neural network.

A classic convolutional neural network was designed with the concept of multi-task learning as the DNN architecture, which is schematically shown in Figure 4-6. The inputs to the DNN are the grayscale maps extracted from the raw CAD data of the target workpiece after springback, and the outputs from the network include predictions of the required loading stroke and two physical properties, true stress and strain, at the regularisation spot before springback. The network includes 6 convolutional layers for feature learning/extraction and 4 fully connected layers for regression. TensorFlow Core v2.2.0 was used as the platform to establish and train the neural networks.

4.4 Data acquisition and training method

4.4.1 Data acquisition

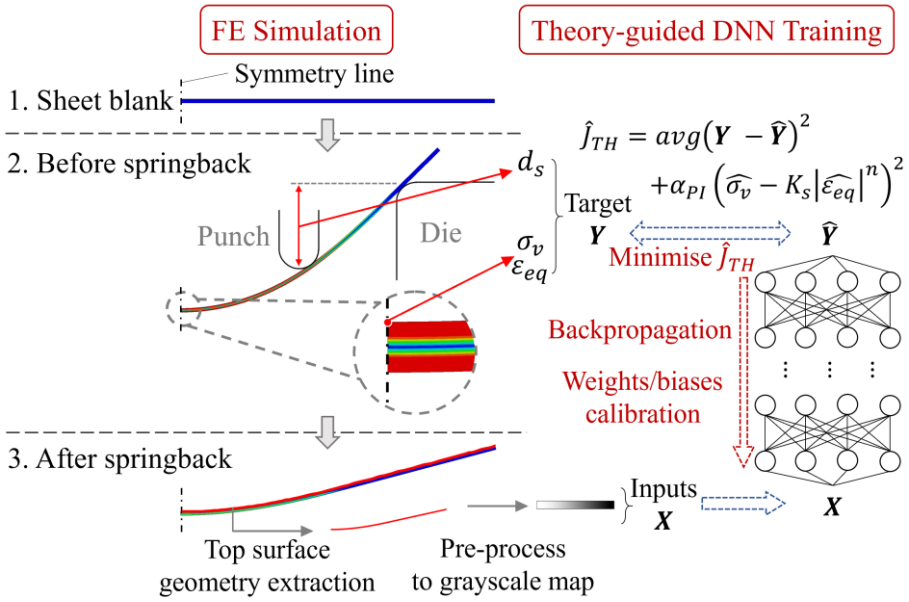


Figure 4-7. Training data acquisition and data flow between FE simulation and TG-DNN training.

Figure 4-7 shows the training data acquisition through the FE simulation and how the data were used for TG-DNN training, which is taking place in the dataflow as shown by the red dashed square in Figure 4-5. The experimental results of loading stroke (d_s) and two physical properties (σ_v and ε_{eq}) at the mid-point of the workpiece top surface were collected before the springback of the workpiece. Thus, only one regularisation spot was set. These data were used as the target data in calculating the physics-guided objective function as shown in Eq. (4-6).

The top surface geometry feature of the workpiece after springback was extracted and pre-processed to grayscale map before it was fed into the network as inputs.

A total number of 201 simulations were performed with loading stroke ranging from 11 to 31 mm with an increment of 0.1 mm. The dataset is in a form of input-output pair, with input as the workpiece geometry representation data and output as the loading stroke. The output also includes physical properties of true stress σ and true strain ε. The whole dataset was then randomly split into a *training set* and *test set*. Three different training to test data ratios were used in this research: 80%/20%, 50%/50% and 20%/80%. This composition of data was designed to evaluate the performance of the TG-DNN and the data-driven DNN with different amount of training data.

4.4.2 Training method

For training the SVR, the radial basis function kernel was used, and the hyperparameters of "epsilon" and "C" were set as 0.1 and 1, respectively. The SVR was established and trained with *scikit-learn* library in Python. The Adam optimisation algorithm [57] was used to train the DNNs, with default parameters of its hyperparameters ($\beta_1, \beta_2, \epsilon$) in TensorFlow Core v2.2.0. The DNNs were trained with mini-batch training method [56]. At training, the training data were shuffled after each training epoch, which denotes one full cycle through the whole training dataset, to reduce the sequence bias to the learning. The sizes of minibatch used in different experiments are presented in Section 4.5.

For optimisations using the gradient decent algorithm, the model parameters updated at each learning step are related to the learning rate multiplying the gradient of objective function over model parameters. Once the model parameters are approaching their global/local optimum, a low learning rate is recommended to facilitate convergence and reduce the wiggling of solution at the end of training. Thus, an exponentially decaying learning rate scheme was used to realise this process. The learning rate decaying scheme is defined as follow:

$$\eta = \eta_0 \cdot {s_d}^{T_d/t_d} \tag{4-7}$$

where η denotes the learning rate at the current decaying time step whose number is t_d, η_0 is the initial learning rate, s_d is the decaying rate and T_d is the total decaying step number. s_d and T_d were set as 0.96 and 100,000 in this research.

The hyperparameters search space for the DNNs studied are presented in Table 4-3. The highest test accuracy that the model can reach in these search domains would be regarded as the best performance the model can achieve.

Table 4-3. Hyperparameters search space

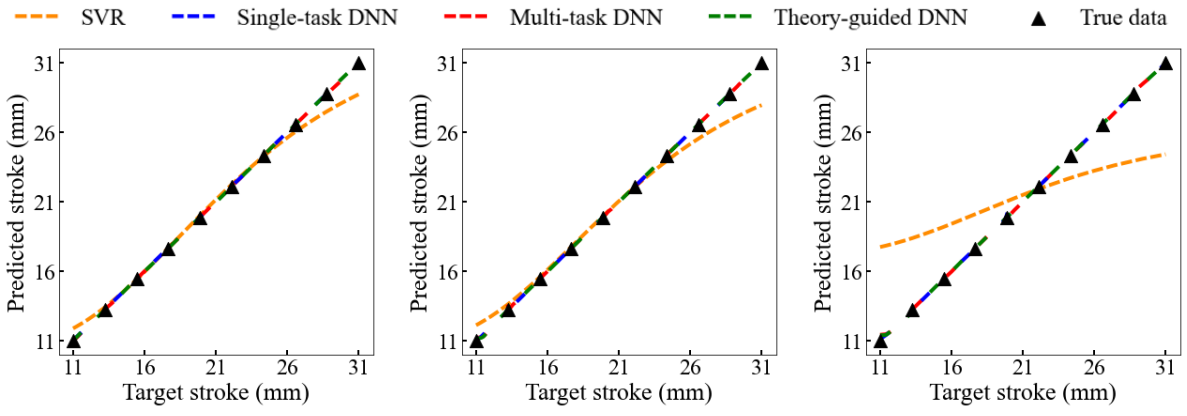
	Data-driven DNN	Theory-guided DNN
Learning rate	$\eta_0 \in [10^{-8}, 10^{-3}]$	$\eta_0 \in [10^{-8}, 10^{-3}]$
Regularisation coefficient	-	$\alpha_{TH} \in [10^{-4}, 1]$

4.5 Learning results

All the neural models were trained with the hyperparameters search space shown in Table 4-3. The model that results in the lowest relative L_2 norm on the test set was then trained for another 3 times, and the generalisation error of the network was obtained by averaging the performance of the three trained models. The average results of $\Delta GEOM_{pred}$ from the three training is used to measure the performance of the machine learning models.

4.5.1 Randomly sampled data

To evaluate the performance of the machine learning models (SVR, data-driven DNN and TG-DNN), the original 201 data were split into three different training-test data composition: 80%/20%, 50%/50% and 20%/80%. The training data were randomly sampled from the whole dataset for 10 times to train the models, of which the average results are shown in Figure 4-8.



(a) 80% train/20% test **(b) 50% train/50% test** **(c) 20% train/80% test**

Figure 4-8. Comparison between the true data and predictions of the data domain from SVR, ST-DNN, MT-DNN and TG-DNN, which were trained with different amount of data. The true data were obtained from four-point bending test of AA6082. The training data were randomly sampled from the whole dataset.

As the theory-guided learning exploits the concept of multi-task learning, to ascertain whether this method improves learning performance, two data-driven DNNs, multi-task DNN (MT-DNN) and single-task DNN (ST-DNN), were established for comparison. The four DNNs used the same architecture as shown in Figure 4-6, except for the ST-DNN which only has one neuron in the output layer. All the machine learning models were applied to predict the loading stroke, based on the target workpiece geometry, in a four-point bending process of a sheet metal with material of AA6082. The number of trainable parameters for ST-DNN is 6,217,769, and that for MT-DNN and TG-DNN is 6,217,791. The huge number of parameters compared to 201 data samples could arouse concern of overfitting, which can often occur in DNN training. However, from the learning results in the following sections, data-driven DNN only overfits when training data is decreased to less than 40.

From Figure 4-8, the prediction accuracy from SVR had apparent deterioration when the amount of training data was reduced, while those from the three neural models had negligible change. Furthermore, the predictions from DNNs conformed well with the true data in all three cases, while the prediction from SVR had clear deviation from its true data, especially when only 20% data were used for training. This could be due to the significantly less trainable parameters of SVR than DNN, for which it cannot learn the intrinsic function when decreasing training data. The results of the maximum distance between the target and predicted workpiece shape ($\Delta GEOM_{pred}$), which is used to measure the prediction accuracy, for the four machine learning models are shown in Table 4-4. From the results in the table, the data-driven models and the theory-guided model showed indiscernibly good performance in all three experiments.

Table 4-4. Comparison of the average of the maximum distance between the target and predicted workpiece shape ($\Delta GEOM_{pred}$) from the well-trained SVR and DNNs evaluated on the total data domain (unit: mm).

Data composition	80% train / 20% test	50% train / 50% test	20% train / 80% test
SVR	0.43 ± 0.05	0.68 ± 0.04	3.21 ± 0.24
Single-task DNN	0.01 ± 0.01	0.02 ± 0.01	0.03 ± 0.01
Multi-task DNN	0.02 ± 0.01	0.03 ± 0.02	0.03 ± 0.01
Theory-guided DNN	0.02 ± 0.01	0.02 ± 0.01	0.03 ± 0.01

4.5.2 Scarce data

Because of the nearly equal performances of the data-driven and TG-DNN, another experiment was designed with new composition of training and test data. The ratio of training to test data was kept to the poorest condition as in last experiment (about 20%/80%), while the

training data were chosen to eschew large continuous regions of data domain which were unseen at training, instead of having a relatively uniform data for training obtained from random sampling. Two data compositions, named ***symmetric*** (includes stroke values in the range of 11-12 mm, 21-22 mm and 30-31 mm) and ***asymmetric*** training data (includes stroke values in the range of 11-15 mm and 30-31 mm), were designed in this study. This design of training data is to better reflect practical situations in industry, where experimental data are expensive and scattered. In addition, to evaluate the learning consistency of neural models, two more experiments were designed to apply both data-driven DNNs and TG-DNN to 1) four-point bending process with a new material of SS400 and 2) air bending process with the same material of AA6082. Thus, the learning capabilities of the neural models were evaluated in experiments with three varying conditions:

1) Different raw data structure: using ***symmetric/asymmetric training data***, which were acquired from the ***four-point bending process*** of the ***AA6082*** workpiece.
2) Different workpiece material: using ***symmetric training data***, which were acquired from the ***four-point bending process*** of the ***SS400*** workpiece.
3) Different forming process (applications): Using ***symmetric training data***, which were acquired from the ***air bending process*** of the ***AA6082*** workpiece.

Figure 4-9 shows the training loss curves for the three DNNs during training and their generalisation performance under four different conditions. The generalisation performance was measured by evaluating the well-trained DNN on the test set, and the predictions for the whole data domain from different models are plotted. It was seen that the training losses of DNNs in different cases all converged to a small value at the end of training. Due to an extra theory-guided regularisation term in the objective function for the TG-DNN, the convergence of which would conflict that of the standard objective function, the converged training loss of the TG-DNN were larger than those of the pure data-driven DNNs. From the prediction curves of the three models, compared to the performance of the DNNs trained with randomly sampled training data in Figure 4-8a, the prediction from the TG-DNN remained a good fit to its target value, while the data-driven DNNs trained with scattered data in this section failed to learn the true function in the unsupervised data domain in most experiments. Besides, the TG-DNN showed a good learning consistency from its consistent performance under various training conditions and in different applications. Table 4-5 shows the average results of $\Delta GEOM_{pred}$

from the DNNs in all experiments, in which the TG-DNN was superior to the two data-driven DNNs. It is worth noting that the MT-DNN outperformed the ST-DNN in most cases, which indicates that the learning of loading stroke could be improved by learning extra tasks of local true stress and strain.

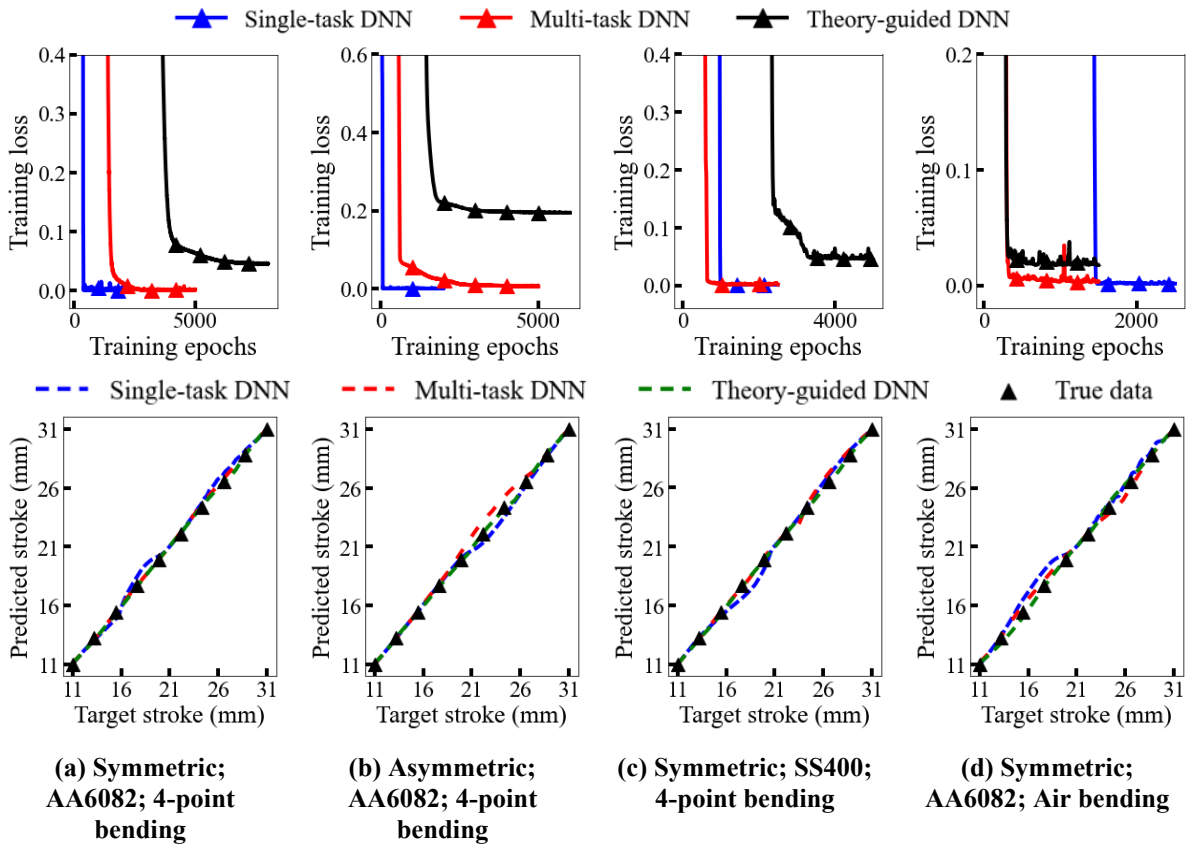


Figure 4-9. The training loss curves for the three DNNs (upper row) and the evaluations of the DNNs on the total data domain (lower row) after training with about 20% training data acquired from different applications. The training data was designed to be symmetric and asymmetric to have large continuous regions of missing data domain.

Table 4-5. Comparison of the average of the maximum distance between the target and predicted workpiece shape ($\Delta GEOM_{pred}$) from the well-trained DNNs evaluated on the total data domain in all experiments.

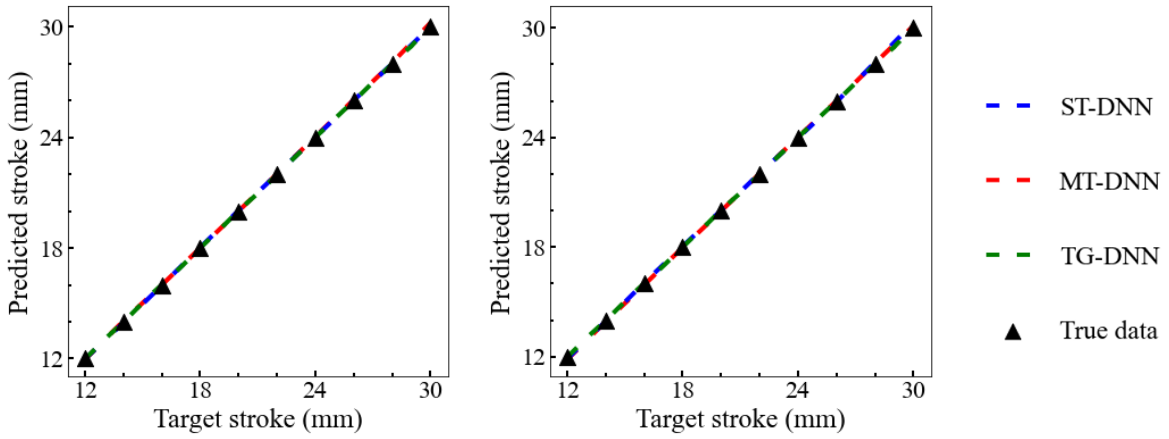
Data structure	Workpiece material	Manufacturing process	ST-DNN	MT-DNN	TG-DNN
Symmetric	AA6082	4-point bending	0.34 ± 0.03	0.09 ± 0.02	**0.08 ± 0.02**
Asymmetric	AA6082	4-point bending	0.26 ± 0.03	0.35 ± 0.04	**0.11 ± 0.02**
Symmetric	SS400	4-point bending	0.35 ± 0.04	0.22 ± 0.03	**0.07 ± 0.01**
Symmetric	AA6082	Air bending	0.44 ± 0.04	0.30 ± 0.05	**0.20 ± 0.03**

4.5.3 Severely scarce data

In industry, experimental data can be rather expensive, for which of great significance is the predictive model to have a good performance with very limited training data. Thus, in this

section, a new experiment was designed to have a total number of 21 examples of dataset, with loading stroke ranging from 11 to 31 mm with unit increment. The whole dataset was then split into a training set and test set, with data pairs with odd numbers of loading stroke (i.e. 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31 mm) as *training set* and data pairs with even numbers of loading stroke (i.e. 12, 14, 16, 18, 20, 22, 24, 26, 28, 30 mm) as *test set*. Thus, there were in total 11 and 10 data examples in training set and test set, respectively. To comprehensively evaluate the performance of the DNNs with scarce data, three different amounts of training data were used for training the neural networks. The original training set with 11 examples was designed to be the "full training set". Likewise, the training data with 6 (with stroke values of 11, 15, 19, 23, 27, 31 mm) and 3 (with stroke values of 11, 21, 31 mm) evenly distributed examples were referred to as the "half training set" and "1/4 training set".

All the networks were trained to convergence with the batch size of 2. Figure 4-10 presents the generalisation results of the ST-DNN, MT-DNN and the TG-DNN, well-trained with full and half original training data, on the test set. The training data were obtained from the four-point bending process of AA6082 workpiece. It can be seen that, with full and half amount of training data, all DNNs performed nearly equally well. This can also be observed in Table 4-6, in which the values of $\Delta GEOM_{pred}$ between the target and predicted workpiece geometry from all models were negligibly small with full and half training data.



(a) Full training data **(b) Half training data**

Figure 4-10. Comparison of generalisation capability on test set between the data-driven and theory-guided DNNs with a) full data b) half data, obtained from four-point bending process of AA6082 workpiece.

Table 4-6. Comparison of the average of the maximum distance between the target and predicted workpiece shape ($\Delta GEOM_{pred}$) from the well-trained data-driven and theory-guided DNNs evaluated on the test set. The data were obtained from four-point bending process of AA6082 workpiece (unit: mm).

	ST-DNN		MT-DNN		TG-DNN	
Training data	Full data	Half data	Full data	Half data	Full data	Half data
Test accuracy	0.03 ± 0.01	0.08 ± 0.01	0.05 ± 0.01	0.08 ± 0.01	0.03 ± 0.01	0.10 ± 0.01

With 1/4 training samples, the training data were also designed to be structurally *symmetric* (includes stroke values of 11, 21 and 31 mm) and *asymmetric* (includes stroke values of 11, 13 and 31 mm) for four-point bending of AA6082. The models were also applied to applications of 1) four-point bending process with a new material of SS400 and 2) air bending process with the same material of AA6082 to comprehensively evaluate the performance and learning consistency of the data-driven DNNs and TG-DNN.

Figure 4-11 shows training loss curves for all the DNNs during training and the generalisation performance of them under four different conditions. The generalisation performance was measured by evaluating the well-trained DNN on the test set (with stroke values of 12, 14, 16, 18, 20, 22, 24, 26, 28, 30 mm).

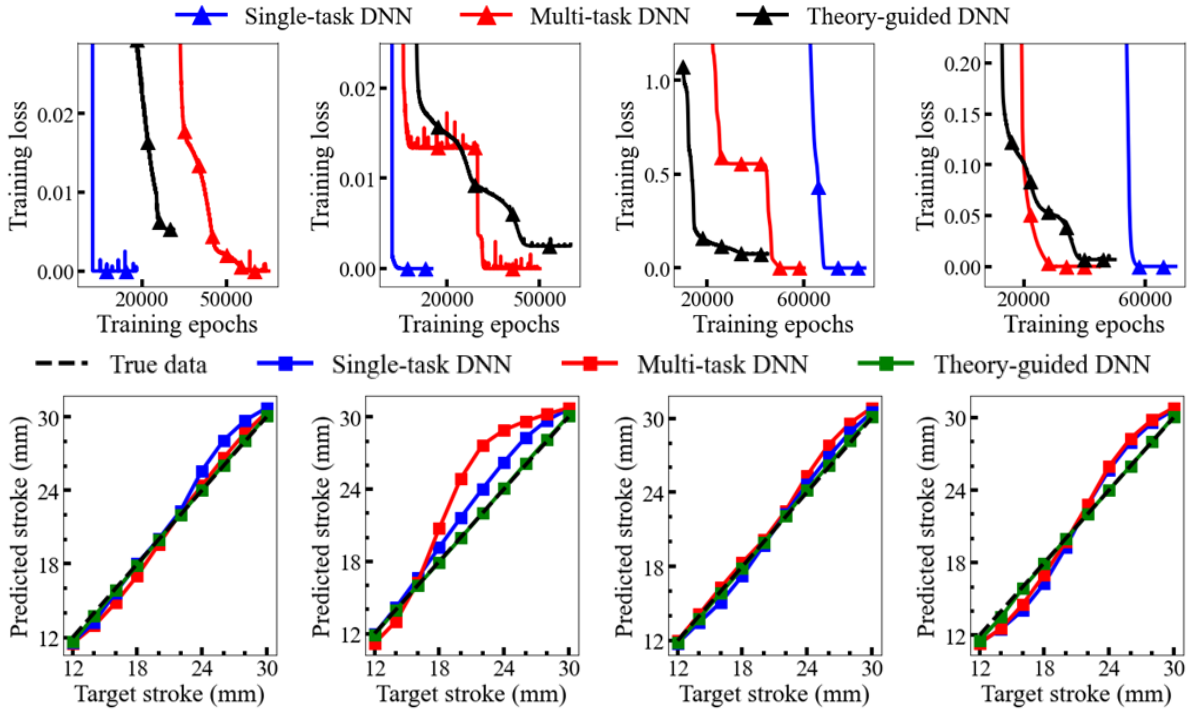


Figure 4-11. The training loss curves for the data-driven and theory-guided DNNs (upper row) and the evaluations of all the DNNs on the test set (lower row) after training with 1/4 training data acquired from different applications.

It was seen that the training losses of all the DNNs in different cases all converged to a small value at the end of training. Similar to the results in experiments with 201 total data shown in Figure 4-9, the TG-DNN had a higher converged loss than the data-driven DNNs due to its extra theory-guided regularisation term in the objective function. From the prediction graphs, the predictions of stroke values on the test set from the TG-DNN had a good match to the target values in all cases. Compared with the multi-task and single-task learning results, where evident generalisation errors were seen, the TG-DNN exhibited effective learning and still held high generalisation performance in the data domain away from the training set. With only 1/4 training data, the data-driven DNN failed to learn the true function in the unsupervised data domain, while the TG-DNN succeeded to retain high learning capability. From the average results of $\Delta GEOM_{pred}$ in different applications shown in Table 4-7, with 1/4 training data, the results of $\Delta GEOM_{pred}$ measured from TG-DNN only had an increase less than 0.1 mm than those obtained from training with full and half data shown in Table 4-6. Thus, the TG-DNN exhibited a significantly more robust and consistent learning capability than data-driven DNNs when applied to problems with different data structures and amount of the training data, different materials of the workpiece and different bending processes.

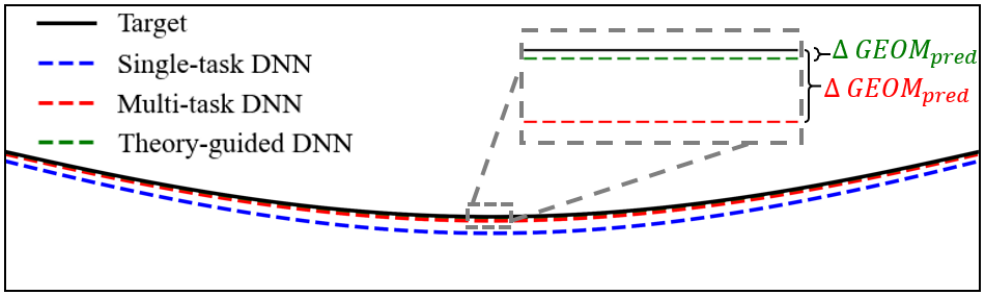
Table 4-7. Comparison of the average of the maximum distance between the target and predicted workpiece shape ($\Delta GEOM_{pred}$) from the well-trained data-driven DNNs and theory-guided DNN evaluated on the test set in all experiments.

Data structure	Workpiece material	Manufacturing process	ST-DNN	MT-DNN	TG-DNN
Symmetric	AA6082	4-point bending	0.80 ± 0.07	0.63 ± 0.06	**0.18 ± 0.02**
Asymmetric	AA6082	4-point bending	1.26 ± 0.10	2.69 ± 0.15	**0.16 ± 0.02**
Symmetric	SS400	4-point bending	0.58 ± 0.06	0.69 ± 0.06	**0.17 ± 0.01**
Symmetric	AA6082	Air bending	1.30 ± 0.09	1.24 ± 0.07	**0.19 ± 0.03**

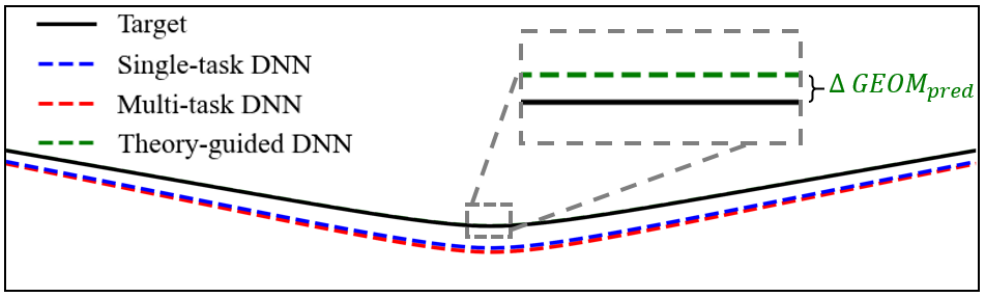
It is worth noting that, in the case with asymmetric training data shown in Figure 4-11b, the TG-DNN also managed to learn the true function in spite of the considerably increased learning difficulties due to large amount of missing data domain for training purpose (between target stroke of 13 and 31 mm). The predictions from both MT-DNN and ST-DNN had evident deviation from the true function in this region. It is well known that the structure of the training dataset has significant influence on the training of neural networks, while the data structure is commonly irregular when acquiring off-the-shelf experimental data. Thus, the introduced TG-DNN, in practice, can effectively alleviate the demand on the amount and structure of expensive experimental data, which could reduce the effort in experiment design. If applied to forming irregular workpieces, empirical equations of calculating stress concentration could be used to

better guide the prediction. Also, from the prediction graphs in Figure 4-9 and Figure 4-11, the performance of the data-driven DNNs trained with only three examples was inferior to that of the DNNs trained with 30-50 data, while the TG-DNN showed a more consistent performance. This indicates the superior learning capability of the TG-DNN than the data-driven DNN trained with very scarce data.

Taking case 1 and case 4 (Figure 4-11a and Figure 4-11d) as examples, with the predicted values for target stroke of 24 mm from the DNNs, the obtained workpiece geometries deformed by four-point bending and air bending were compared with their targets, as shown in Figure 4-12. It can be seen that the geometry predicted from TG-DNN was evidently closer to the target than those from ST-DNN and MT-DNN.



(a) Four-point bending

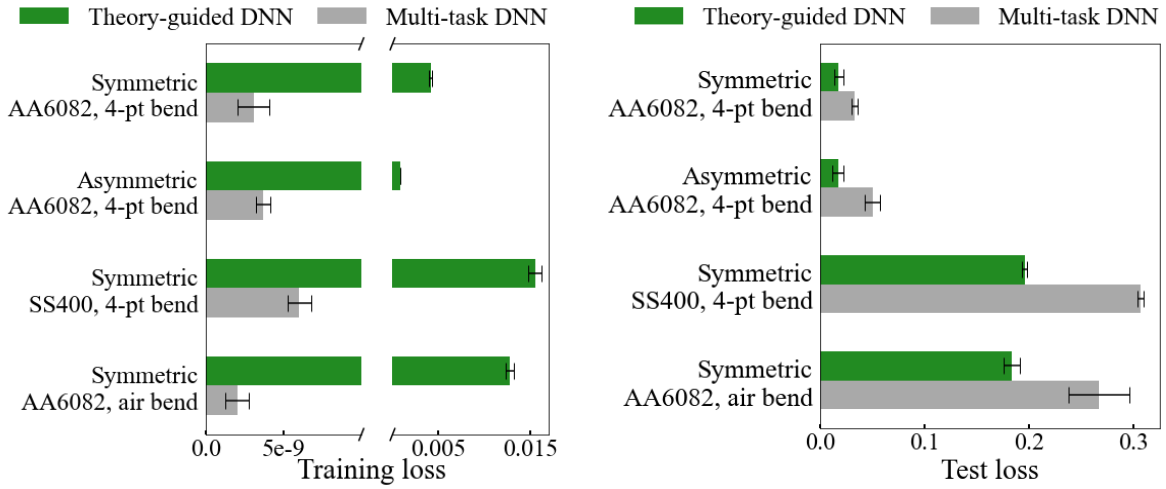


(b) Air bending

Figure 4-12. Comparison between the shape of the target workpiece (with target stroke of 24 mm) and the workpiece manufactured with the stroke predicted by the data-driven and theory-guided DNNs, which were trained with 1/4 symmetric data acquired from mid-point of the AA6082 workpiece from (a) four-point bending and (b) air bending.

To ascertain whether the generalisation accuracy of the DNN was promoted by the proposed theory-guided regularisation method using Swift's law, the final training loss and test loss of the physical properties (σ_v and ε_{eq}) from the well-trained TG-DNN and the MT-DNN in all experiments were compared as shown in Figure 4-13. It can be seen that the training losses of these parameters from data-driven DNN were considerably smaller than those from the TG-DNN, which is due to the convergence conflict between the standard term and the theory-guided

term in the objective function for the TG-DNN. However, the losses of the physical properties on the test set from the TG-DNN were smaller than those from the data-driven DNN, which indicates its better learning outcome in the physical-parameters data domain. This corroborates the learning improvements introduced by the TG-DNN, as the physical properties were directly regularised in the theory-guided objective function. Consequently, the theory-guided regularisation method was demonstrated to improve DNN performance. With Swift's law for training regularisation, the interpretability of the DNN was improved.



(a) Training loss of physical properties **(b) Test loss of physical properties**

Figure 4-13. Comparison between the training and test loss of physical properties (σ_v and ε_{eq}) between well-trained MT-DNN and TG-DNN with 1/4 symmetric training data acquired from mid-point of the AA6082 workpiece from 4-point bending.

4.6 Summary

A novel theory-guided regularisation method for DNN training has been proposed, which utilises Swift's law as guidance to regularise the multi-task learning and improve the learning performance. This method has been compared with a traditional machine learning model, SVR, and a data-driven DNN in learning the intrinsic relationship between the deformed workpiece geometry after springback and the forming parameter under various experimental conditions.

The neural networks, either trained in data-driven or theory-guided, exhibit blatantly better learning capabilities than a traditional machine learning model, SVR, in predicting the loading stroke of a four-point bending and an air bending process. With less training data (from 201 to 30-50 samples), the performance of the SVR deteriorates sharply, while those of the neural models have negligible decrease.

The TG-DNN outperforms the conventional pure data-driven DNN for its superior generalisation accuracy, when only scarce and scattered experimental data (30-50 samples) are available for training. When decreasing the amount of training data to a very small value (3 samples), the performance of the data-driven DNNs suffers while that of the TG-DNN retains consistent.

With varying experimental conditions, including the structure and amount of training data, workpiece material and sheet metal forming application, the TG-DNN exhibits a more robust learning capability and learning consistency than the data-driven DNNs.

Other than bending, The TG-DNN could also be applicable to the sheet metal forming processes, including flexible hammering processes, where the physical properties (σ_v and ε_{eq}) are, in general, monotonically related to the forming parameters. The theory-guided learning also improves the interpretability of the traditional data-driven DNNs.

Chapter 5

Multiple punches: tool path planning with deep supervised learning technology

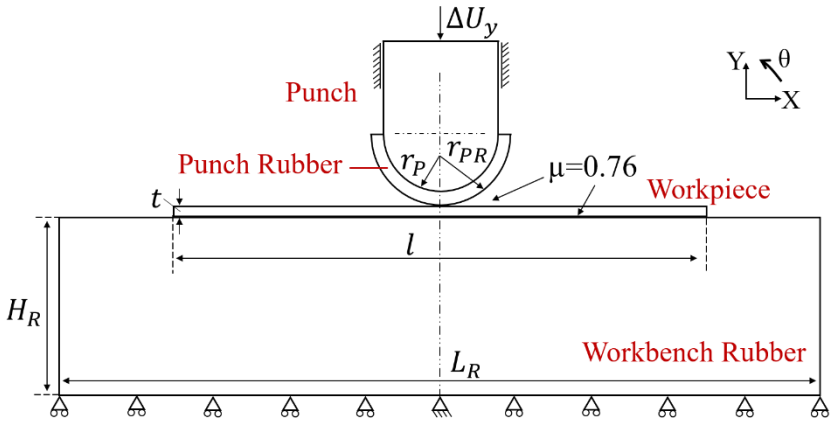
5.1 Introduction

Tool path planning strategy is of the most importance to configure the Process Generation Module in the intelligent sheet metal forming system as shown in Figure 3-2. Although many studies have been reported on machine learning applications to metal forming, few has focused on tool path prediction of a consecutive sheet metal stamping.

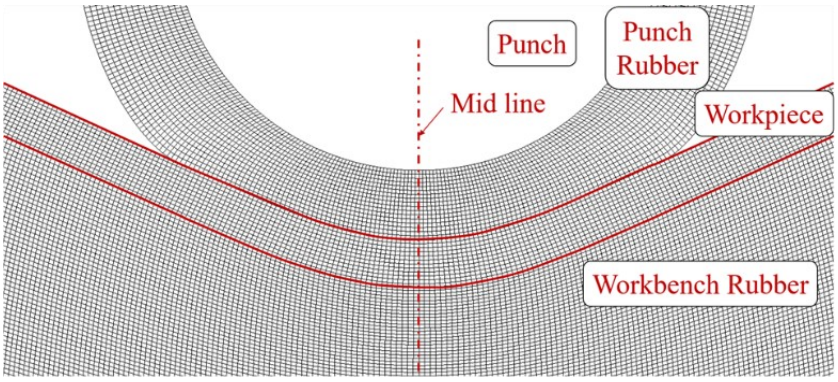
This chapter presents a recursive tool path prediction framework, impregnated with a deep learning model, for the forming sequence planning of a consecutive rubber-tool forming process. This work has been presented at Journal of Materials Processing Technology [158]. The deep learning model recursively predicts the forming parameters, namely punch location and punch stroke, for each deformation step, which yields the optimal tool path. Three series of state-of-the-art deep learning models, namely single feature extractor, cascaded networks and LSTM models were investigated and compared. Section 5.2 presents a novel consecutive forming process, followed by its digitisation method and data acquisition design for model training in Section 5.3. Section 5.4 and 5.5 introduce the architecture of the recursive prediction framework and the details of the three series of deep learning models, respectively. Finally, Section 5.6 presents the performances of these models in tool path planning, and the misprediction from the best model is further evaluated in Section 5.7.

5.2 Forming experiments test setup

In this section, a rubber-tool forming process is proposed to deform sheet metals so that the product has a Class A surface condition. Figure 5-1 shows the test setup, FE model and FE mesh of the rubber-tool forming. The workpiece, laying on a large piece of silicon rubber, was deformed by a rubber-wrapped rigid punch. During forming, the workpiece was only in contact with rubber and thus protected from scratches and dents. As shown in Figure 5-1a, the punch was only allowed to translate along Y-direction. The bottom of the workbench rubber was supported by a rigid body and its mid-point was fixed in all directions. The forming experiments were performed computationally on Abaqus/2019, with mesh shown in Figure 5-1b. The punch was assured to deform the workpiece at the expected location by ensuring the coincidence of three lines: the mid lines of the punch rubber and workbench rubber and the line passing through the punch location in thickness direction of the workpiece. The specification of the test setup in Figure 5-1a is presented in Table 5-1.



(a) The test setup and FE model



(b) The FE mesh and the alignment during the rubber-tool forming process

Figure 5-1. The test setup, plane strain FE model and FE mesh in forming test.

Table 5-1. Specification of the test setup of rubber-tool forming.

Features	Workpiece length, l	Workpiece thickness, t	Punch radius, r_P	Punch rubber radius, r_{PR}	Workbench Rubber length, L_R	Workbench Rubber height, H_R
Value (mm)	100	1	5	7	160	60

For simplicity, the dimension in the third direction was assumed infinite and therefore the numerical experiments were performed in 2-D space. The workpiece and the rubbers were meshed with 8-node biquadratic plane strain quadrilateral elements with reduced integration, CPE8R. The punch was meshed as a rigid body. The workpiece and two pieces of rubber were set to be homogeneous. The friction coefficient between the workpiece and rubber was 0.76 [159]. To determine the mesh size of the components in FE simulation, a grid-independence study was performed. Based on this study, a total of 103,716 mesh elements were used.

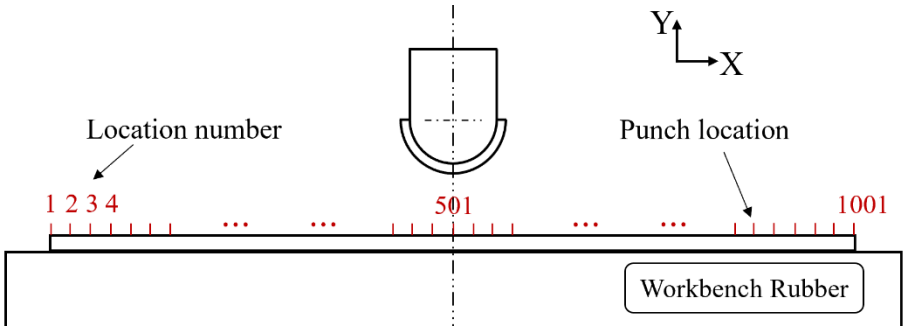
The materials of the workpiece and rubber used in this research was AA6082 [150] and natural rubber [160]. The properties of these materials used for simulation are shown in Table 4-2 and Table 5-2.

Table 5-2. Material properties.

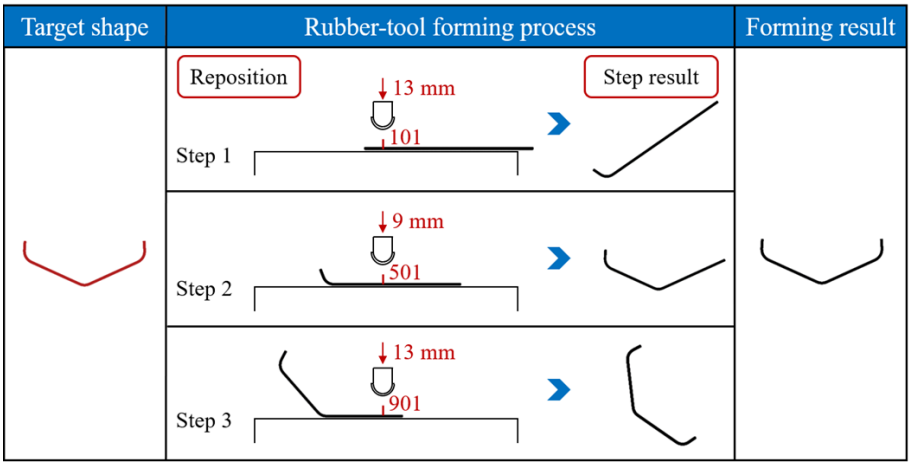
Materials	Young's modulus, E (GPa)	Poisson's ratio, γ_p (-)	Density, ρ (g/cm3)
Natural rubber	0.0025	0.48	0.92

To realise the consecutive forming process, of great importance is the role automatic repositioning plays. For the ease of geometry data processing and assurance of forming flexibility, 1001 equidistant discrete locations on the upper surface of the workpiece were selected as possible punch locations, which is the same number as the mesh nodes on this surface in computational domain. Figure 5-2a shows a schematic diagram of all the selectable punch locations during forming. The punch locations were numbered from left to right, starting from 1 to 1001. The denotation of deformed workpiece is in the form of "location-stroke" for one single punch and "location1-stroke1_location2-stroke2_..._locationN-strokeN" for multiple punches. For example, "501-5mm" denotes a single punch at location 501 with stroke of 5 mm; "301-5mm_501-9mm" denotes a sequential deformation with stroke of 5 mm at location 301 and 9 mm at location 501. Figure 5-2b shows the rubber-tool forming process, in which the flat sheet was sequentially deformed to its target shape. During the forming process, the deformed workpiece at each step was reserved as the workpiece shape for the next forming step. The workpiece was repositioned over the workbench rubber to assure that the punch

location was aligned with the central axis of the punch. The repositioning of sheet metal during forming was realised by the alignment of three lines as shown in Figure 5-1b. At the start of each consecutive forming operation, the punch and workbench rubber were reset to their initial positions while the workpiece was rotated and translated for alignment.



(a) Schematic diagram for 1001 punch locations



(b) The rubber-tool forming process for deforming the sheet metal to its target shape ("101-13mm_501-9mm _901-13mm" in this example)

Figure 5-2. Punch locations and rubber-tool forming process.

5.3 Process digitisation and data acquisition

It is common knowledge that deep neural networks (DNNs), especially the convolutional neural networks (CNNs), have achieved significant success in localising and extracting prediction-related features from image-based data as suggested by [83]. In this context, rather than feeding the original coordinates data of workpiece from the simulations into machine learning models, these data were transformed into binary pixel-based images. Figure 5-3 shows the data representations of workpiece before and after pre-processing. Based on the workpiece

size, the size of the binary image was 288×576, of which the spatial distance between two contiguous pixels was 0.2 mm.

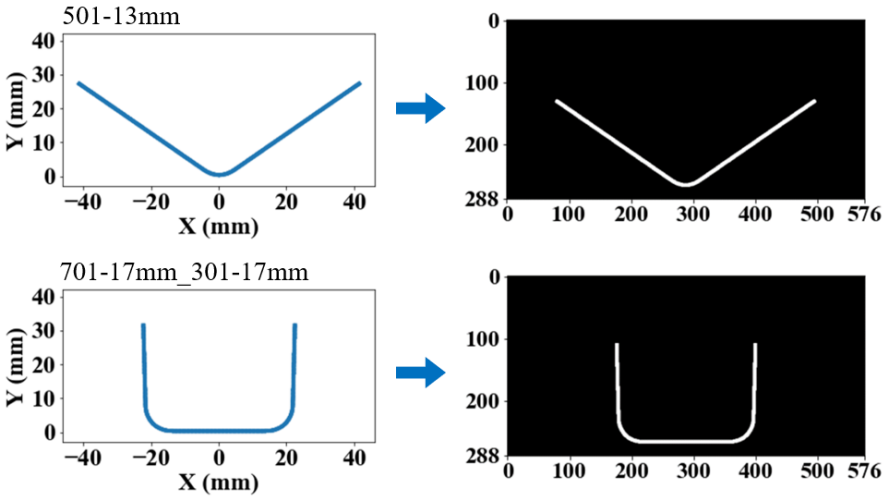


Figure 5-3. Examples of workpiece representations in graph drawn by coordinates data (left) and binary image (right).

The pre-processing algorithm is presented in Algorithm 1.

Algorithm 1: Workpiece representation pre-processing algorithm

1. Create a zero matrix M of size 288×576 (precision of 0.2)
2. Create a mapping function f_m converting spatial coordinates to pixel position: $i = 196 - 5y, j = 5x + 288$, where (x,y) is the spatial coordinates and (i,j) is the pixel position
3. Create a round function f_r: $x \leftarrow (x \,/\, 0.2) \cdot 0.2$, where / denotes integer division
4. **for** mesh node n = 1, N **do**
5. **If** node n is on the outer surface of the workpiece **then**
6. Find the four pixels around node n with f_m and f_r: $p(i_1,j_1)$, $p(i_2,j_2)$, $p(i_3,j_3)$, $p(i_4,j_4)$
7. Find the pixel closest to node n: $p(i_k,j_k)$
8. $M(i_k,j_k) = 1$
9. **else**
10. Find the four pixels around node n with f_m and f_r: $p(i_1,j_1)$, $p(i_2,j_2)$, $p(i_3,j_3)$, $p(i_4,j_4)$
11. $M(i_1,j_1) = M(i_2,j_2) = M(i_3,j_3) = M(i_4,j_4) = 1$
12. **end if**
13. **end for**
14. Export the binary image matrix M

In machine learning, unlike classification which predicts a label presented in the dataset, regression predicts a continuous quantity which could not be in the training data. In real cases, the selection of punch location and stroke on workpiece can be continuous, for which the prediction of them is considered a regression problem. The machine learning model is expected

to capture the intrinsic relationship in the causality between the selection of punch location/stroke and the resulting workpiece shape. For this reason, a series of FE simulations were performed with five selectable punch locations and corresponding punch strokes. Figure 5-4 shows all the punch locations and corresponding punch strokes selected in the simulations, which also indicates the structure of the dataset. In the simulations, the sheet metals were sequentially deformed at the selectable punch locations with selectable strokes (up to 3 combinations of location and stroke).

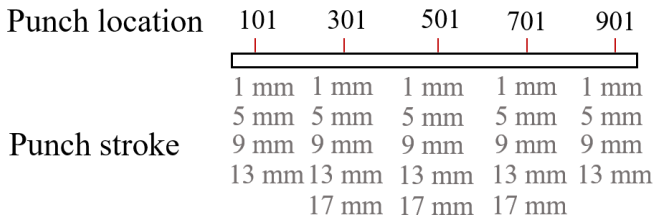
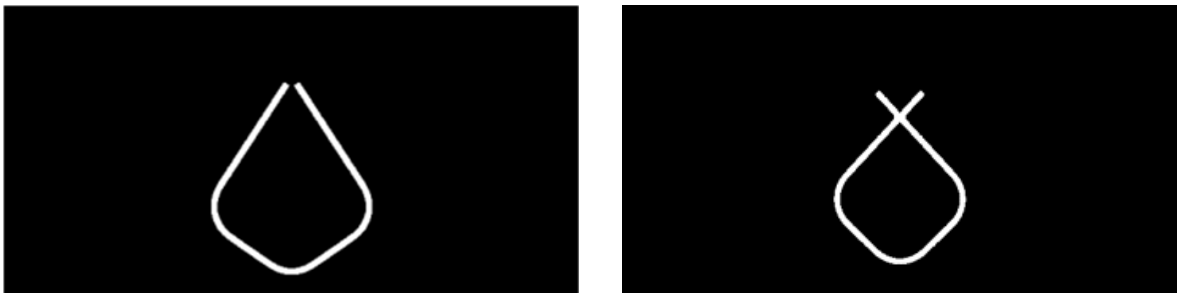


Figure 5-4. Illustration of possible punch locations and stroke options.

Every possible deformation result was collected with the selectable punch locations and strokes, and useful results were sifted for model training. First of all, infeasible workpiece shapes, which have conflicts with real manufacture, were removed from the dataset. Figure 5-5 shows two examples of infeasible workpiece shapes. One of them has insufficient opening between the workpiece ends to leave room for the punch, and the other workpiece intersects with itself which is not practical.



(a) 301-17mm_501-13mm_701-17mm **(b) 301-17mm_501-17mm_701-17mm**

Figure 5-5. Infeasible workpiece shape with insufficient opening for punch in (a) and self-intersection in (b).

After excluding all the infeasible results, the number of geometries for workpieces deformed with one, two and three punches are 23, 422 and 5737. The models were trained with two datasets, respectively termed as Dataset within 2 and 3 punches, consisting of the images of the workpiece deformed by maximum of 2 (including 1 and 2) and 3 (including 1, 2 and 3)

strokes. Dataset within 2 punches includes 445 workpiece geometries and the one within 3 punches includes 6182 geometries.

The 6182 data were reallocated and assembled to conform to different model architectures. As the deep learning models were used for learning a regression problem, the training data cannot include one-to-many data. i.e., the same input features but different labels, which would cause the value of prediction to lie between these labels. Although a target workpiece could be deformed through different tool paths, only one of them was reserved in the training data. Thus, when configuring the training data, the extra data causing one-to-many problem were trimmed. The training data were trimmed in the way that only the label with the lowest value of punch location was reserved when one-to-many problem appeared. For example, for target workpiece "101-13mm_501-9mm _901-13mm", the labels for workpiece "101-13mm" include "501 and 9mm" and "901 and 13mm". Thus, the label of "901 and 13mm" was removed from the dataset. However, there were still training data with feature of "101-13mm_901-13mm" and label of "501 and 9mm" for data augmentation.

5.4 Consecutive tool path prediction framework and its training method

In real workshop-scale sheet metal forming process, the tool path planning can determine the delivery quality of the final product. During forming, it is a common case that one is prone to take the next forming operation at the location where the local shape of the current deformed workpiece has salient differences from its target. For example, when making a metal bowl, blacksmiths would put the sheet metal on a sandbag and start the strike from the middle part of the metal. In this regard, the data pipeline through the deep learning model was developed to respect this heuristic. Figure 5-6 presents the data pipeline through the deep learning models at training phase and application phase. It is noted that the inputs-outputs pair describes a single step through the whole tool path prediction process, of which the three words highlighted in red are the denotations signifying temporal characteristics. The "current" workpiece is the most recently obtained workpiece shape after deformation, and "target" workpiece introduces itself literally. "Immediate" punch location/stroke signifies the forming parameters of the punch on the "current" workpiece.

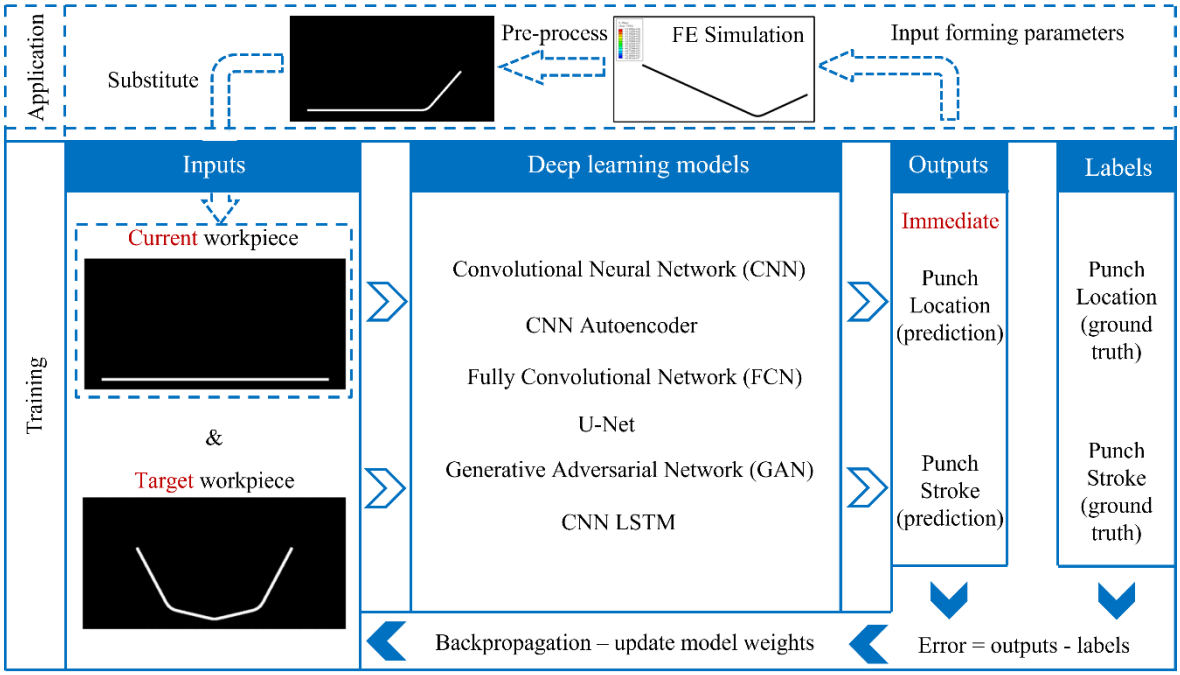


Figure 5-6. Data pipeline through deep learning models at the training phase (solid square) and application phase (dashed square). The filled arrows denote the data flow during the training of the deep learning model.

The dataset for training and evaluating the deep learning models consisted of data in the format of the inputs-labels pair. At the training phase, for each training step, the predicted forming parameters were obtained by feeding the inputs data from the training dataset into the deep learning model. The error matrix between predictions and labels was computed and used for updating the deep learning model parameters through backpropagation.

At the application phase, the well-trained models were used to predict the tool path of a new "target" workpiece. At each forming step, the prediction of the "immediate" forming parameters was used to perform an FE simulation. The new workpiece geometry, retrieved from the simulation, was then pre-processed to be a binary image. The pre-processed geometry substituted the "current" workpiece in the first channel of inputs, which was then concatenated with the "target" workpiece and fed into the deep learning models again for new "immediate" punch parameters. The recursive prediction would yield the tool path from blank sheet metal to its desired geometry, once the "current" workpiece shape conforms to the "target" workpiece to a prescribed extent.

The deep learning model $\mathcal{M}$ in Figure 5-6 can be defined as follow:

$$\mathcal{M} = \hat{\boldsymbol{Y}}(\boldsymbol{X}; \boldsymbol{\theta}) \tag{5-1}$$

where $\widehat{\boldsymbol{Y}}(\boldsymbol{X};\boldsymbol{\theta})$ is the prediction from the deep learning model, which denotes the predictions of punch location and stroke. $\boldsymbol{X} \in \mathbb{R}^{h\times w\times d}$ is the concatenated binary image input to the network model, of which h and w denote the height and width of the binary image while d denotes the number of images concatenated. For example, in the example shown in Figure 5-6, h, w and d are 288, 576 and 2, respectively. $\boldsymbol{\theta}$ represents trainable parameters, i.e., *weights* and *biases*, and hyperparameters, such as learning rate.

As introduced above, the tool path prediction was configured as a regression problem. Therefore, MSE, which measures the Euclidean distance between prediction from the model and the ground truth as shown by Eq. (2-3), was used as the objective function for training the deep learning models. The number of learning task was 2. An exponentially decaying learning rate scheme was used as in Section 4.4.2.

In this research, the networks were trained using Keras with TensorFlow v2.2.0 as backend on a facility with the specification of NVIDIA Quadro RTX 6000 GPU, 24 GB of RAM memory. All deep learning model in this study were trained with the Adam optimisation algorithm. Default values of hyperparameters ($\beta_1, \beta_2, \epsilon$) of Adam in Keras were used.

The prediction accuracy of punch location and stroke is measured as follow:

$$accuracy = (1 - \left|\frac{\hat{y} - y}{y}\right|) \times 100 \tag{5-2}$$

where $\hat{y}$ is the predicted value and y is the ground truth value. It is noted that if the prediction accuracy is calculated to be a negative value, i.e. the prediction is two times away from its ground truth, the accuracy is set to be zero.

5.5 Deep learning models

For tool path prediction, several deep learning models have been investigated in this research, which can be separated into two groups: models able to process spatial features, such as CNNs, and those able to process temporal features such as LSTMs. In the former category of models, in addition to using a single feature extractor, cascaded networks were also developed to improve the learning performance. The cascaded networks include an image reconstruction network and a convolutional net, which generate the next deformed workpiece shape before predicting the "immediate" forming parameters. Figure 5-6 shows the image

reconstruction model studied in this paper, including CNN autoencoder, fully convolutional network (FCN), U-Net and generative adversarial network (GAN).

5.5.1 Single feature extractor

There are two CNNs investigated in this research as a single feature extractor, including a simple classic CNN developed in Section 4.3 and a state-of-the-art feature extractor, VGG16 [89], with only minor changes of the last layer to adapt to the data pipeline as shown in Figure 5-6. The feature extractors were trained to minimise the MSE loss between the feature extractor outputs and their ground truth values. The details of the network architectures of both models can be found in corresponding papers.

5.5.2 Cascaded networks

Figure 5-7 shows the data pipeline through the cascaded model, and the overall data pipeline can be obtained by substituting the cascaded model into the architecture shown in Figure 5-6.

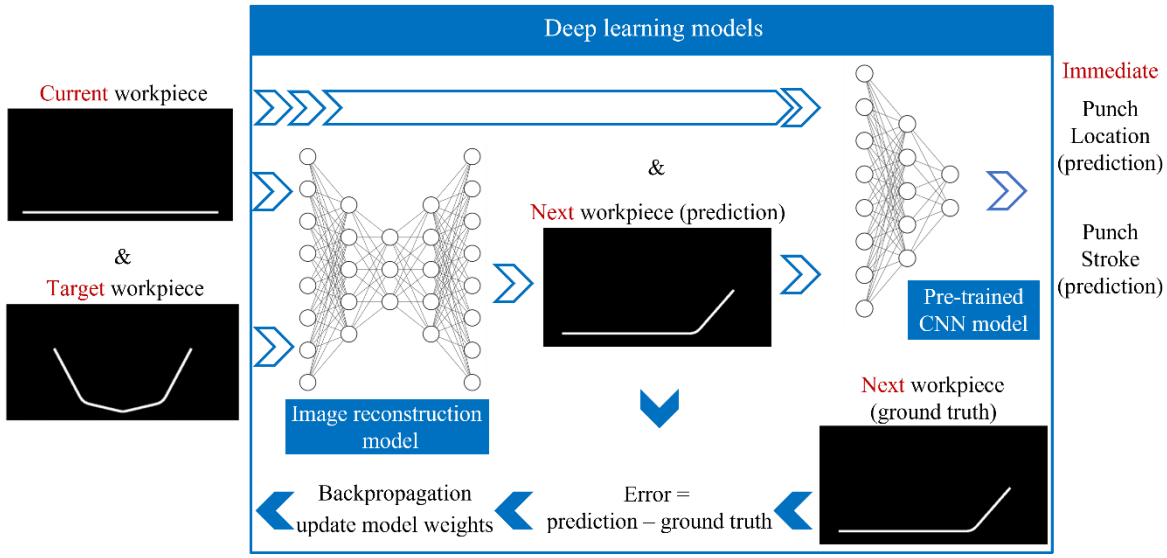


Figure 5-7. Data pipeline through the cascaded network. A target workpiece "301-9mm_501-5mm_701-9mm" is used as an example for illustration. The filled arrows denote the data flow during the training of the image reconstruction model. The top long unfilled arrow denotes that the "current" workpiece is also an input to the CNN model. The CNN model is pre-trained in prior and implemented as a feature extractor in the cascaded network.

The cascaded model is composed of an image reconstruction model and a feature extractor (CNN), of which the CNN model is pre-trained with the dataset of current-next workpiece binary image pair. The reason of pre-training the CNN with different training data from those used for training the single feature extractor is illustrated in the results Section 5.6.1.1. A target workpiece "301-9mm_501-5mm_701-9mm" is used as an example for illustration. The image reconstruction model takes the "current" and "target" workpiece images as input data to generate "next" workpiece image, which denotes the prediction of workpiece shape after a single punch from the "current" workpiece. The predicted "next" workpiece is then concatenated with the "current" workpiece (indicated by the long unfilled arrow in Figure 5-7) and fed into the CNN model for the prediction of the "immediate" punch location and stroke. In this research, the image reconstruction models investigated include convolutional autoencoder (CAE), FCN, U-Net and GAN. VGG16 was used as the CNN model in the cascaded network because of its superior performance compared with the classic CNN, which will be discussed in Section 5.6.1.1.

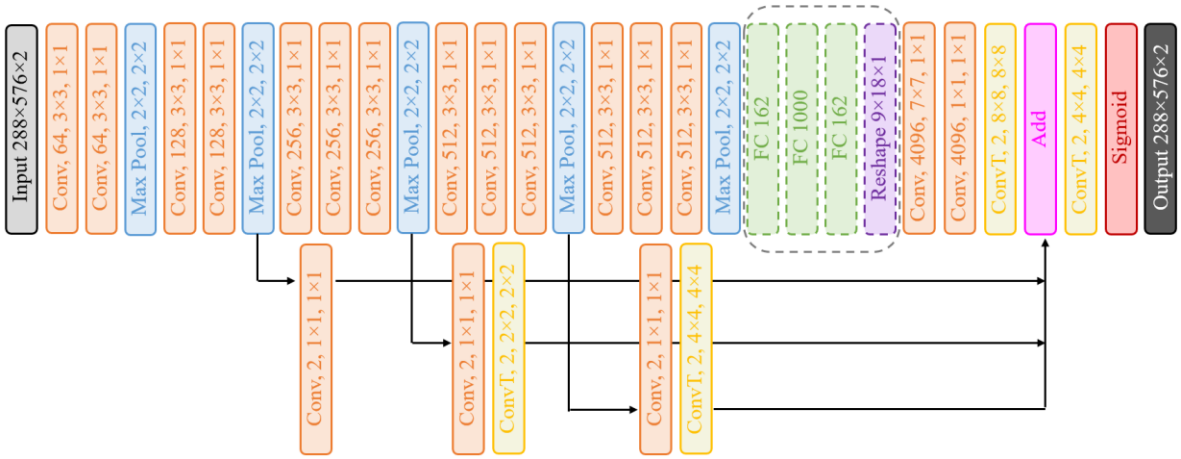
5.5.2.1 *The architectures of image reconstruction models*

There are six image reconstruction models investigated in this experiment: CAE, FCN4s [161], semi-FCN4s, U-Net [162], Res-SE-U-Net [163] and cGAN [164]. The architectures of the six models and details of parameters are shown in Figure 5-8. In terms of the selection of FCN, instead of using FCN8s which was stated to outperform other FCNxs models in semantic segmentation according to [161], FCN4s was selected in this study as it was found, unlike FCN8s, to be able to capture and predict the minimum workpiece curvature deformed by 1 mm stroke. As shown in Figure 5-8b, a model named "semi-FCN4s", which added 1000 neurons as the latent space before upsampling, was developed to exploit the prominent feature learning and semantic segmentation ability of the CNN autoencoder and FCN4s, respectively. Res-SE-U-Net is a special format of U-Net with several Res-SE blocks between the encoder and decoder blocks, which has been found to have salient performance in engineering problems as suggested by Nie et al. [163] and Zhou et al. [138]. In this work, six Res-SE blocks were used with a downscale ratio of 16 in each block. cGAN, unlike all the previous generative models, includes another discriminator model to classify whether the image generated by the generator model is "real" or "fake". It is noted that only the generator in the cGAN was used as the image reconstruction model in the cascaded model. As shown in Figure 5-8d, the generator has the

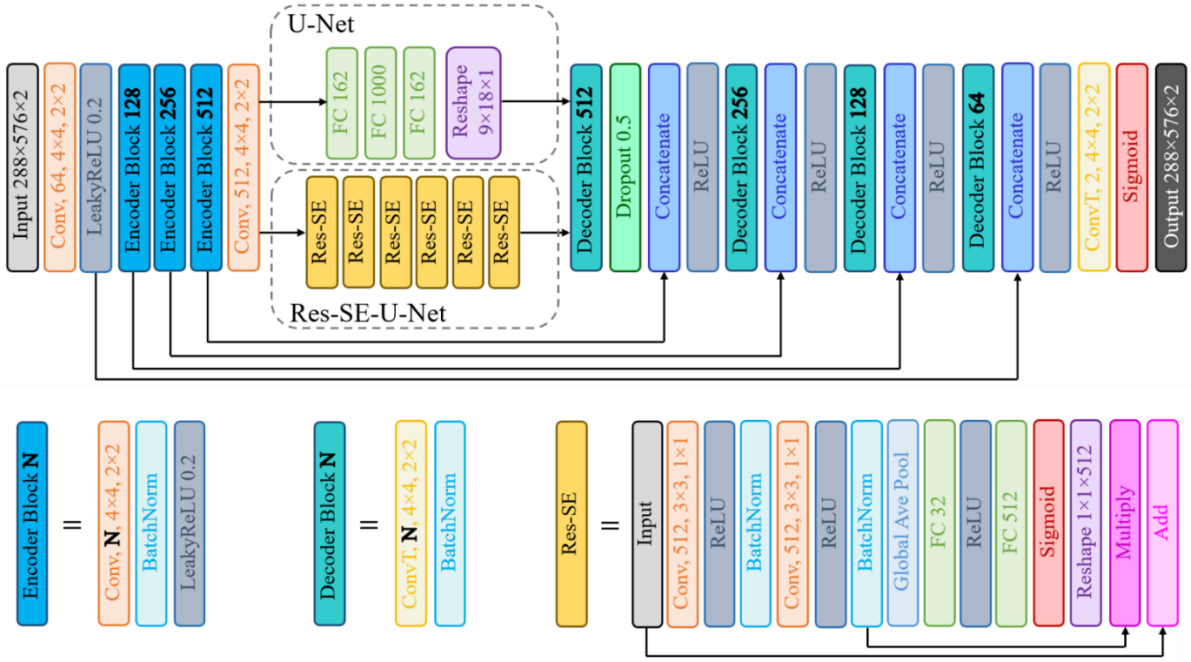
same network architecture with the U-Net. Thus, the two models differ only in the training process.



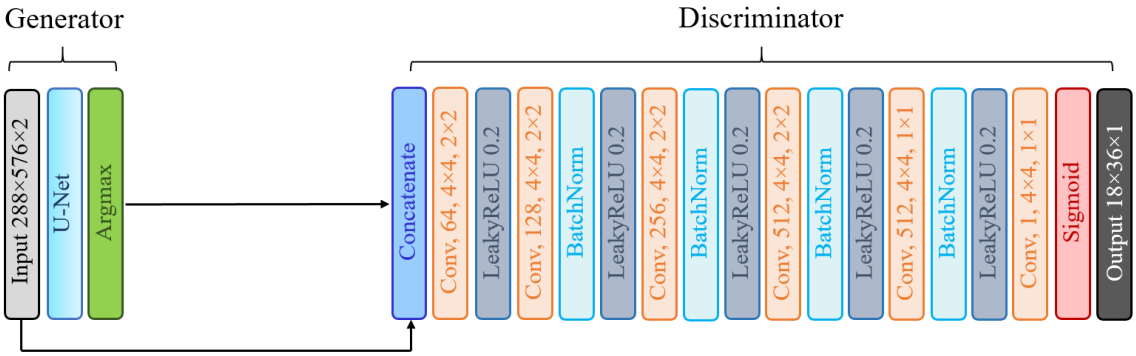
(a) CAE



(b) FCN4s (without layers in dashed square) and semi-FCN4s (with layers in dashed square)



(c) U-Net (with the layers in the upper dashed square only) and Res-SE-U-Net (with the layers in the lower dashed square only)



(d) cGAN (with convolutional "PatchGAN" classifier as the discriminator)

Figure 5-8. Network architectures and details of parameters of the six image reconstruction models used in the cascaded network. Conv and pooling layers are defined in the format of (channels, kernel, stride) and (pool, stride), respectively. ConvT denotes transposed convolution layer. In all architectures, all convolutional layers are followed by a ReLU activation layer, which is not denoted in figures for brevity. "Same padding" is used in conv and convT for all models. As the learning is configured as a classification problem, the outputs of the models have two channels.

5.5.2.2 *Training of the cascaded network*

Since a pre-trained CNN is used in the cascaded model as a feature extractor without further training, the training of the cascaded network equals the training of the image reconstruction model. As one of the investigated image reconstruction models, FCN model, is known to be superior in performing end-to-end and pixel-to-pixel classification, the "next" workpiece prediction is configured as a classification problem for all image reconstruction models for conformity. In addition, the performances of other models in learning classification and regression problems were compared, which have negligible differences and will not affect the qualitative analysis.

Two classes were created, with "0" denoting the black background pixels in the image and "1" denoting the workpiece pixels. A pixel-wise binary cross-entropy function [56] was used as the objective function in training the image reconstruction models, which can be simplified and defined as follow:

$$J_{cross-entropy}(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{Y}) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{c} y_i^k \log\left(\hat{y}_i^k\right) \tag{5-3}$$

where n denotes the number of pixels, c denotes the number of classes.

Unlike CAE, FCN and U-Net, GAN has a different training process. In the training of GANs, the input image was classified as "real" if it was from the dataset or "fake" if it was generated from the generator. The discriminator was trained directly with input features and labels, while the generator was updated via the discriminator. The two models were simultaneously trained in an adversarial process until the prediction from the generator could fool the discriminator. In this research, a conditional generative adversarial network (cGAN) was developed, which was trained with the objective function suggested by Isola et al. [164]. The cGAN consists of a convolutional "PatchGAN" classifier as the discriminator, which classified if each $M \times N$ patch in the input image was real or fake. The patch size in this research was 18×36. In practical training, the discriminator and the generator were updated with the loss function of binary cross-entropy and mean absolute error (MAE), respectively.

5.5.3 LSTM models

LSTM was initially developed by Hochreiter and Schmidhuber [72] to deal with the vanishing gradient problem which can frequently occur during the training of a vanilla recurrent neural network (RNN). The vanishing gradient problem can incur data loss from previous time history and even impede the neural network from further training when the network becomes deep.

Inspired by the model architecture used for image captioning developed by Vinyals et al. [165], which took an image and a partial caption sequence as inputs to predict the next word describing the image, the model used in this experiment also had a many-to-one configuration. Figure 5-9 shows the data pipeline through the deep learning model in this research when the LSTM model was used, and the overall data pipeline can be obtained by substituting the LSTM model into the architecture shown in Figure 5-6. The deep learning model took two inputs, target workpiece image and the partial forming sequence images, to predict the immediate punch location and stroke. The workpiece images were encoded to vectors with dimension of 256 by a feature extractor before fed into the LSTM or other operations. The LSTM took input sequence of four time steps, each of which was an encoded vector of the workpiece shape image in the deformation history. If the deformation history contained less than four time steps, the rest of the vectors in the input sequence was zero padded. Thereafter, the output from the LSTM

was added to the encoded vector of the target workpiece and fed into a feed forward network. The LSTM model was trained to minimise the MSE objective function.

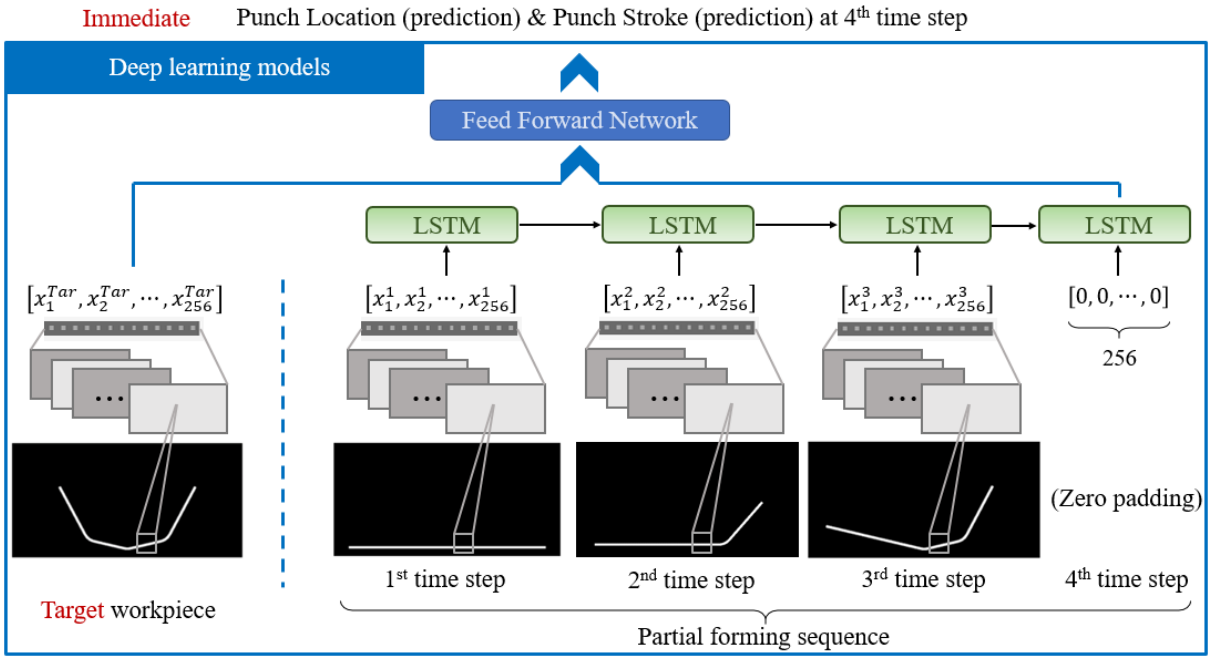


Figure 5-9. Data pipeline through the CNN LSTM. A target workpiece "301-9mm_501-5mm_701-9mm" is used as an example for illustration. The input includes the workpiece shapes at the first three deformation time steps and the model predicts the forming parameters at the fourth time step.

Figure 5-10 shows the architectures of the two LSTM models used in this research.

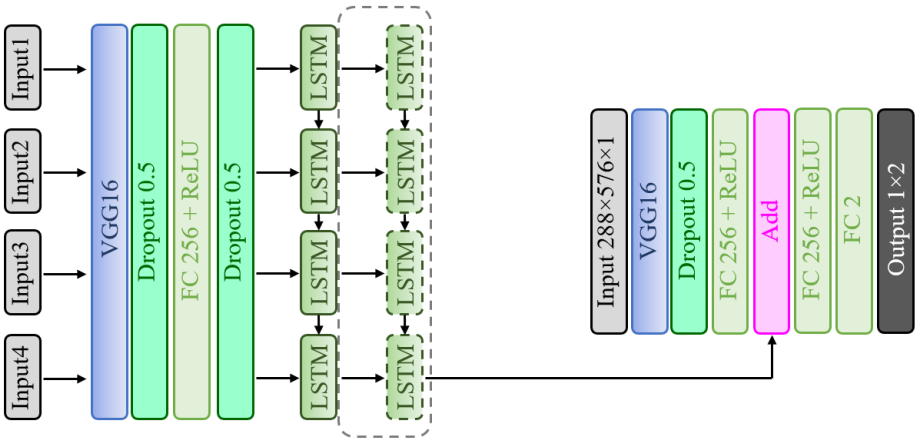


Figure 5-10. The architecture of single LSTM model (without layers in dashed square) and stacked LSTM model (with layers in dashed square). The four inputs at four time steps to the LSTM has dimension of 288×576×1. The LSTM layer has 256 units

The LSTM layer had 256 units, which took inputs from four time-steps. The number of LSTM units is not shown in the figure for brevity. The input to the LSTM was a sequence of binary images showing workpiece shapes at each forming step, and the input followed by a

feature extractor before adding the outputs of the LSTM was the target workpiece image. VGG16, which was trained with the LSTM, was used as the feature extractor for feature extraction from the input images, and the two VGG16 models followed by two inputs shared the same weights and biases. Since VGG16 was used for spatial feature extraction from the inputs, the model is also named CNN LSTM.

5.6 Learning results

Figure 5-11 shows the t-SNE feature representation for Dataset within 3 punches (with 6182 data). t-SNE is a dimensionality reduction technique, which is widely used for visualising the correlation between high dimensional data [166]. From Figure 5-11, it can be seen that the workpieces of similar shapes were in the same cluster, and the dataset was compact without massive data loss.

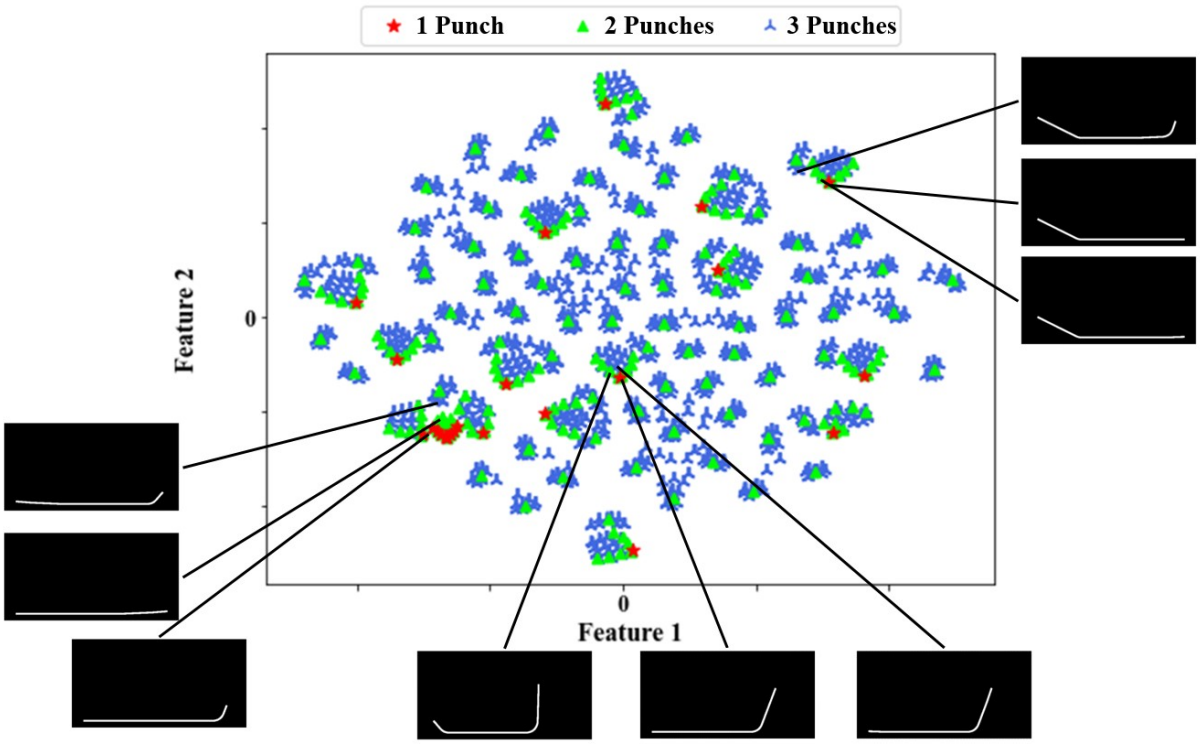


Figure 5-11. t-SNE feature representation for Dataset within 3 punches.

All the neural models were trained with their hyperparameters search space and the model with the lowest loss on the test set was then trained for another 3 times. The prediction accuracies from the three training were averaged and used to measure the performance of these models.

5.6.1 Dataset within 2 punches

5.6.1.1 Single feature extractor

For tool path prediction, two CNNs, a simple classic CNN and VGG16, were first used for prediction of the forming parameters at each forming step. The key training parameters are shown in Table 5-3.

Table 5-3. Key parameters for CNN training with Dataset within 2 punches.

Parameter	Total data	Training data	Test data	Batch size	Initial learning rate, η_0
Value	656	590	66	32	[1e-5, 1e-2]

Table 5-4 summarises the predictions of tool path from the classic CNN and VGG16. It can be seen that overfitting occurred in the training of both CNNs because of the around 10% difference between the prediction results on training and test dataset, respectively. In addition, for both models, the test loss was more than 10 times greater than the training loss. Although VGG16 slightly outperformed the classic CNN in the evaluations on both training and test dataset, the two models did not fulfil the industrial precision requirement in terms of their low prediction accuracies. The poor performances of both CNNs indicate that a single feature extractor model is incapable of predicting the tool path from "current" workpiece to "target" workpiece, based on topology learning.

Table 5-4. Learning results from both CNN models trained with Dataset within 2 punches on the prediction of tool path.

Model	Train			Test		
	Loss	Location accuracy (%)	Stroke accuracy (%)	Loss	Location accuracy (%)	Stroke accuracy (%)
Classic CNN	0.034	92.38	84.83	0.4	83.19	73.25
VGG16	0.016	92.98	90.63	0.38	86.93	79.92

To further evaluate the CNNs learning performance, they were trained to predict the forming parameters for a single punch without forming sequence. The inputs data to the models, shown in Figure 5-6, were redesigned to replace the "target" workpiece with "next" workpiece. "next" workpiece denotes the workpiece geometry deformed from the "current" workpiece with a single punch, whose forming parameters were predicted by the model. In this context, both classic CNN and VGG16 were trained with the new dataset without forming sequences. Table 5-5 presents the key parameters used in the training.

Table 5-5. Key parameters for training with the new dataset without forming sequence.

Parameter	Total data	Training data	Test data	Batch size	Initial learning rate, η_0
Value	445	400	45	32	[1e-5, 1e-2]

Table 5-6 summarises the prediction of punching parameters without forming sequences from the classic CNN and VGG16. Compared with the training results shown in Table 5-4, the performance of both models trained with dataset without forming sequences blatantly outweighed those trained with dataset including forming sequences by around 10%, and both training and test losses were reduced by 10 times. In this regard, a single feature extractor model had promising performance in topology learning between successively deformed workpieces. In addition, VGG16 outperformed the classic CNN in generalisation accuracy, by about 5%, on predictions for both punch location and stroke.

Table 5-6. Learning results from both CNN models trained with new dataset without forming sequence.

Model	Train			Test		
	Loss	Location accuracy (%)	Stroke accuracy (%)	Loss	Location accuracy (%)	Stroke accuracy (%)
Classic CNN	0.0032	97.51	95.68	0.0409	92.18	90.87
VGG16	0.0016	98.79	97.64	0.0246	**98.18**	**95.15**

5.6.1.2 Cascaded networks

As discussed in Section 5.6.1.1, a single feature extractor is insufficient for the tool path prediction. The increasing development of deep learning in image-to-image translation or image reconstruction has boosted its applications in areas such as satellite map translation [167], image semantic segmentation [168] and medical image synthesis [169]. In this paper, a cascaded network was developed to cascade an image reconstruction model and a CNN as the deep learning model in the recursive prediction framework shown in Figure 5-6. The cascaded network architecture is shown in Figure 5-7.

To evaluate the accuracy of predictions from the image reconstruction models, intersection over union (IoU) was used to measure the similarity between the prediction and its ground truth image. IoU for workpiece prediction (class of 1) is defined as following:

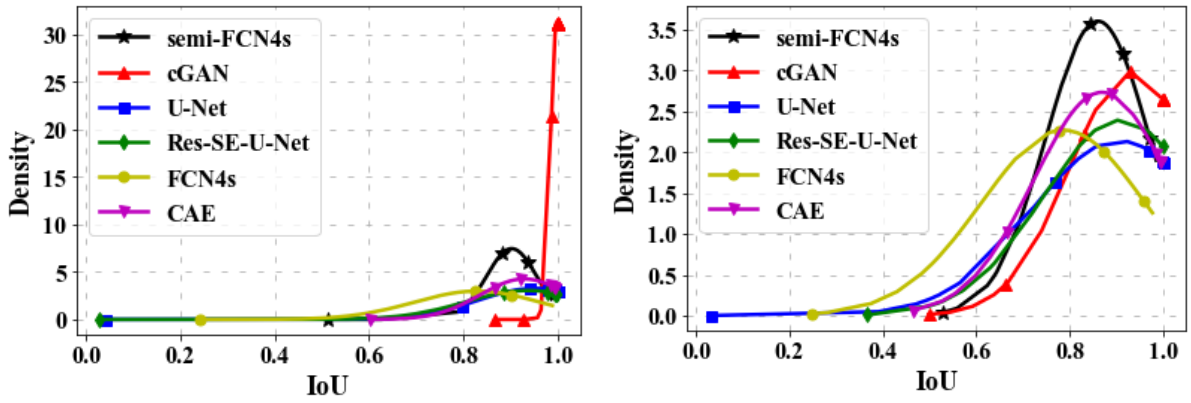
$$IoU = \frac{target \cap prediction}{target \cup prediction} = \frac{TP}{FN + TP + FP} \tag{5-4}$$

where *target* and *prediction* denote the pixel set of the target and predicted workpiece images. *TP*, *FP* and *FN* denote *True Positive*, *False Positive* and *False Negative*, respectively. Table 5-7 shows the key parameters used for training all six image reconstruction models.

Table 5-7. Key parameters for training image reconstruction models with Dataset within 2 punches.

Parameter	Total data	Training data	Test data	Batch size	Initial learning rate, η_0
Value	656	590	66	32	[1e-5, 1e-3]

To evaluate the performance of the six image reconstruction models, the IoU score density distributions over its whole score domain from these models were compared in Figure 5-12. It shows that the cGAN, which had the highest mean IoU score on both training and test dataset, outperformed the other five models in learning the workpiece structural transformation pattern during deformation. It is noted that the semi-FCN4s model, which was designed to incorporate FCN4s and CAE models, had a higher mean IoU score than FCN4s and the same score with CAE. The semi-FCN4s also had a higher IoU density than both FCN4s and CAE. In other words, the semi-FCN4s was superior to either FCN4s or CAE model in image reconstruction. This entails the potential of incorporating FCN and CAE model to acquire prominent semantic plausibility and in image reconstruction logic of the prediction results. The U-Net and Res-SE-U-Net were found to have almost identical performance on training set, while the Res-SE-U-Net slightly preceded the U-Net in generalisation as its predictions with high IoU scores were more concentrated.



(a) Training set **(b) Test set**

Figure 5-12. The IoU results distribution over its whole score domain from six image reconstruction models.

The performance of the cascaded model was evaluated by cascading the pre-trained image reconstruction model and the VGG16 model as shown in Figure 5-7. With the same training

and test dataset above, the prediction accuracies on forming parameters at each forming step of the six cascaded models are presented in Table 5-8. It is blatant that the performance of the cascaded model with cGAN preceded that of the other five models, with only slightly deteriorative generalisation accuracy on stroke prediction than the U-Net and Res-SE-U-Net. Compared with the path prediction accuracy with a single feature extractor VGG16 shown in Table 5-4, the cascaded models with FCN4s and CAE had even worse performance, and most of their prediction accuracies were under 80%. semi-FCN4s, although developed to improve the learning ability, only had similar performance to a single VGG16. However, cascaded models with cGAN, U-Net and Res-SE-U-Net preceded the single VGG16 in generalisation for stroke prediction by about 5%. In addition, cGAN had the best performance on training-set learning among all models investigated so far.

Table 5-8. The prediction accuracy on forming parameters from the cascaded models embedded with six different image reconstruction models with Dataset within 2 punches.

Cascaded Model	Train		Test	
	Location accuracy (%)	Stroke accuracy (%)	Location accuracy (%)	Stroke accuracy (%)
CAE	77.89	67.22	75.99	67.17
FCN4s	79.29	84.07	74.86	78.49
semi-FCN4s	91.48	90.25	86.08	79.73
U-Net	92.54	91.89	86.51	**86.45**
Res-SE-U-Net	87.95	88.63	84.36	85.08
cGAN	**97.39**	**96.88**	**86.70**	84.10

5.6.1.3 LSTM models

The training data for LSTM models were prepared following the rules introduced in Section 5.3. In this context, the key parameters for training both single and stacked LSTM models are summarised in Table 5-9.

Table 5-9. Key parameters for training LSTM models with Dataset within 2 punches.

Parameter	Total data	Training data	Test data	Batch size	Initial learning rate, η_0
Value	1101	880	221	8	[1e-5, 1e-3]

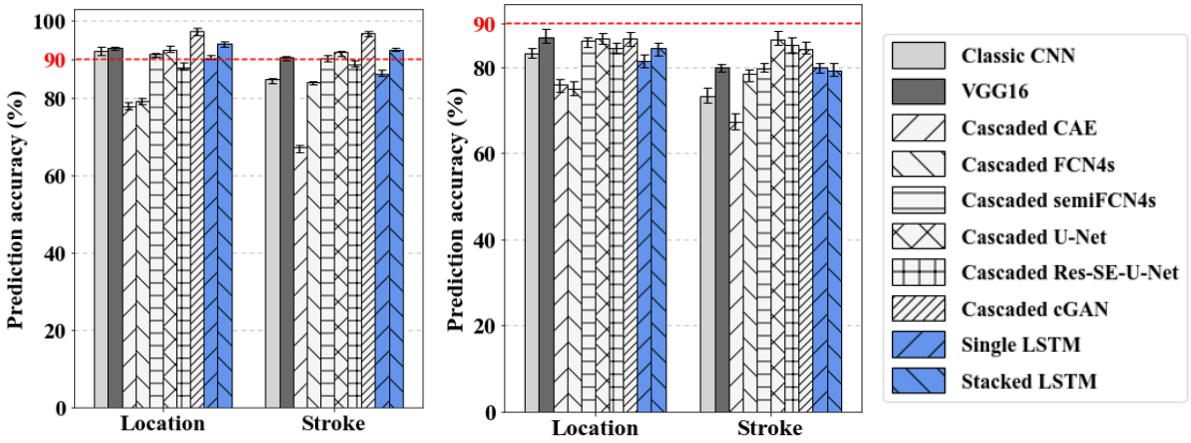
The learning performances of the single and stacked LSTM models are summarised in Table 5-10. It can be seen that the stacked LSTM, which has one more layer of LSTM than the single LSTM model, slightly outperformed the single LSTM. The performance of the stacked LSTM preceded the single one on training-set learning and the generalisation in location

prediction by about 5% and 3%, respectively. Compared with the performance of the cascaded models shown in Table 5-8, both LSTMs had inferior prediction accuracy than the cascaded models with U-Net and cGAN on both training and test dataset.

Table 5-10. Learning results from single and stacked LSTM models trained with Dataset within 2 punches on the prediction of tool path.

Model	Train			Test		
	Loss	Location accuracy (%)	Stroke accuracy (%)	Loss	Location accuracy (%)	Stroke accuracy (%)
Single LSTM	0.141	90.36	86.34	0.310	81.40	80.03
Stacked LSTM	0.069	94.21	92.55	0.268	84.29	79.11

Figure 5-13 shows the comparison of the performance between all investigated models which were trained with Dataset within 2 punches. From the learning results of all deep learning models trained with Dataset within 2 punches, the cascaded model with cGAN as the image reconstruction model was found to have the best learning capability for tool path planning.



(a) Training set **(b) Test set**

Figure 5-13. Bar chart of prediction accuracy from all models on training set and test set of the Dataset within 2 punches, respectively. The learning results are averaged from all three trainings and the error bar shows the performance variation. All trainings are provided with the same amount of raw data in the original workpiece image dataset, and these data are reallocated and assembled to conform to different model architectures.

5.6.2 Dataset within 3 punches

To further evaluate the deep learning models in predicting the workpiece of more complex shape, a new dataset was created with additional data of workpiece deformed by 3 punches. All the deep learning models were trained again with the new dataset and the learning results of them are compared in this section.

5.6.2.1 Single feature extractor

Table 5-11 presents the key parameters for training the single feature extractor with Dataset within 3 punches.

Table 5-11. key parameters for training the single feature extractors with Dataset within 3 punches.

Parameter	Total data	Training data	Test data	Batch size	Initial learning rate, η_0
Value	13084	10467	2617	8	[1e-5, 1e-3]

Table 5-12 shows the learning results from the two CNN models in predicting the tool path of workpiece deformed within 3 punches. It can be seen that, unlike the models trained with Dataset within 2 punches shown in Table 5-4, the single feature extractors can hardly learn the true function of the recursive prediction of tool paths, with the prediction accuracies of about only 70% in all evaluations. Consequently, a single feature extractor was not qualified for tool path prediction of more complex sheet metal workpiece.

Table 5-12. Learning results from single feature extractors trained with Dataset within 3 punches on the prediction of tool path.

Model	Train			Test		
	Loss	Location accuracy (%)	Stroke accuracy (%)	Loss	Location accuracy (%)	Stroke accuracy (%)
Classic CNN	0.207	71.39	70.02	0.812	71.41	69.03
VGG16	0.193	72.69	68.54	0.853	72.84	67.98

5.6.2.2 Cascaded networks

The pre-trained VGG16 trained with Dataset within 2 punches, whose performance is shown in Table 5-6, was evaluated with the Dataset within 3 punches. The prediction accuracies on punch location and stroke were 96.33% and 92.55%, respectively. Thus, it was also used as

the feature extractor in the cascaded network in this experiment. Table 5-13 Table shows the key parameters for training the cascaded models with Dataset within 3 punches.

Table 5-13. Key parameters for training the cascaded models with Dataset within 3 punches.

Parameter	Total data	Training data	Test data	Batch size	Initial learning rate, η_0
Value	13084	10467	2617	8	[1e-5, 1e-3]

From the learning results in Section 5.6.1.2, there were four cascaded models which had equal or better learning performance on tool path prediction than the single feature extractor: semi-FCN4s, U-Net, Res-SE-U-Net and cGAN. These models were trained with the Dataset within 3 punches and their performance are summarised in Table 5-14. Similar to the single feature extractor, the cascaded models also failed to learn the intrinsic function of optimal tool path parameters for a target workpiece, with the best generalisation accuracy of merely 76% on location prediction from cGAN. After increasing the forming complexity from 2 to 3 punches, the prediction accuracies from cascaded models on training and test dataset descended by about 20% and 10%, respectively.

Table 5-14. Learning results from cascaded models trained with Dataset within 3 punches on the prediction of tool path.

Cascaded Model	Train		Test	
	Location accuracy (%)	Stroke accuracy (%)	Location accuracy (%)	Stroke accuracy (%)
semi-FCN4s	72.16	69.12	69.75	66.15
U-Net	74.46	72.89	75.37	72.69
Res-SE-U-Net	74.84	72.21	74.97	71.23
cGAN	75.26	73.31	76.25	73.47

5.6.2.3 LSTM model

Table 5-15 presents the key parameters for training the single and stacked LSTM models with Dataset within 3 punches.

Table 5-15. key parameters for training the LSTM models with Dataset within 3 punches.

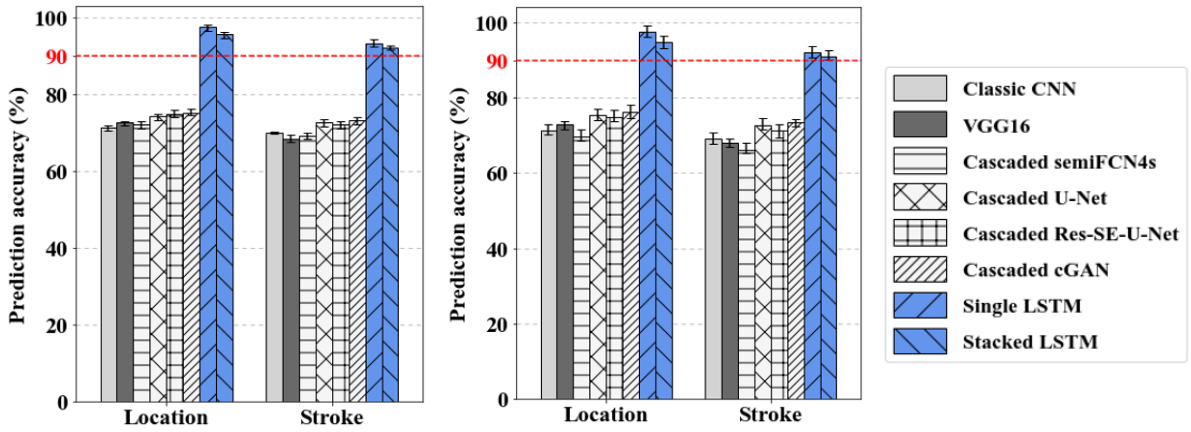
Parameter	Total data	Training data	Test data	Batch size	Initial learning rate, η_0
Value	16397	13117	3280	8	[1e-5, 1e-3]

The learning results of the two LSTM models trained with Dataset within 3 punches are shown in Table 5-16. It can be seen that both LSTM models were blatantly superior to all other models discussed above. In addition, the generalisation losses from these models were barely

slightly higher than the training loss, which indicates that negligible overfitting occurred during the training process. The prediction accuracies of both models evaluated on training and test set exceeded 90%. It is also worth noting that, unlike the LSTM models trained with Dataset within 2 punches, the performances of both LSTMs were significantly improved by over 10% in generalisation ability. Besides, the performance of the LSTM model deteriorated with one more stacked layer of LSTM cell by about 2% in location prediction and 1% in stroke prediction, respectively.

Table 5-16. Learning results from LSTM models trained with Dataset within 3 punches on the prediction of tool path.

Model	Train			Test		
	Loss	Location accuracy (%)	Stroke accuracy (%)	Loss	Location accuracy (%)	Stroke accuracy (%)
Single LSTM	0.051	**97.74**	**93.34**	0.059	**97.63**	**92.20**
Stacked LSTM	0.063	**95.67**	**92.24**	0.076	**94.98**	**91.04**



(a) Training set **(b) Test set**

Figure 5-14. Bar chart of prediction accuracy from all models on training set and test set of the Dataset within 3 punches, respectively. The learning results are averaged from all three trainings and the error bar shows the performance variation. All trainings are provided with the same amount of raw data in the original workpiece image dataset, and these data are reallocated and assembled to conform to different model architectures.

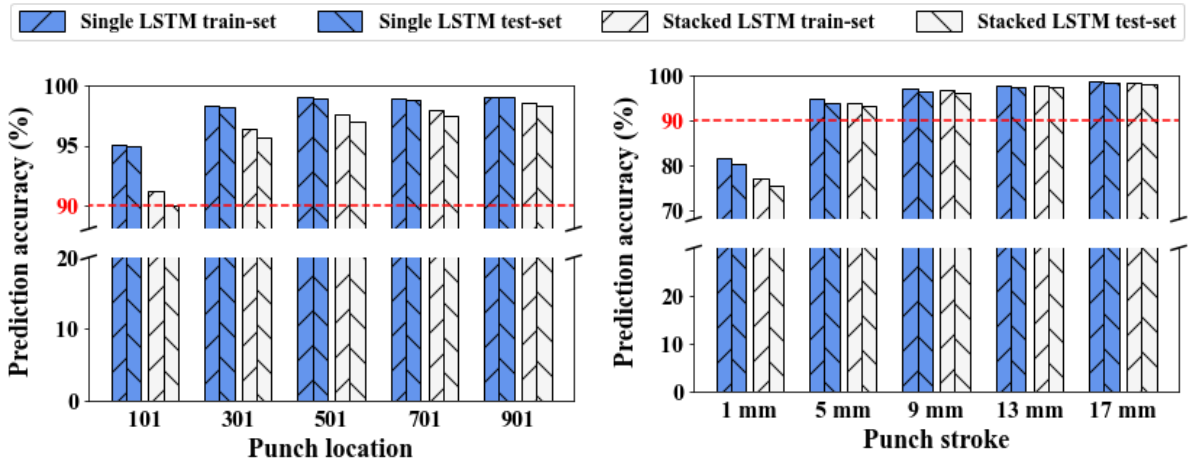
Figure 5-14 shows the comparison of the performance between all investigated models which were trained with Dataset within 3 punches. From the learning results of all deep learning models trained with Dataset within 3 punches, the single LSTM was found to have the best

learning capability for tool path planning. When the path learning becomes complex, it was hard for single feature extractor and cascaded network to learn the path only based on workpiece shape differences. With ability of learning additional temporal features, LSTM outperformed all other models.

5.7 LSTM performance analysis

To further analyse the performance of the LSTM models, the prediction accuracies on each punch location and stroke were obtained for misprediction analysis. Both single and stacked LSTMs were analysed and the predictions for different locations and strokes are compared in Figure 5-15.

Figure 5-15a shows that the prediction accuracies for location 101 were the worst among 5 locations, with just over 95% and 90% from the single and stacked LSTM, respectively. This could be due to that the data with label of location 101 accounted for more percentage than that with other locations in the whole dataset. This phenomenon was caused in data acquisition stage, as introduced in Section 5.3 about the data trimming, lower location number was prioritised to be reserved in the dataset at data trimming. As each training data represented a unique forming scenario, the learning complexity for this location was greater than other locations. This problem could be avoided by dataset structure design in the future.



(a) Location prediction accuracy **(b) Stroke prediction accuracy**

Figure 5-15. The training and test accuracy on each punch location and stroke from single and stacked LSTM models. The accuracies were averaged from all trainings.

From Figure 5-15b, it is evident that, except for stroke 1 mm, the prediction accuracies for other strokes were above 90%. The prediction accuracies for stroke 1 mm were only around 80% and 75% from the single and stacked LSTM, respectively. The misprediction for stroke 1 mm could be due to the minor deformation stroke of 1 mm can make. It was much harder for models to discern the difference between the workpiece shape before and after a deformation with stroke of 1 mm than those deformed by higher strokes. This problem could be prevented by adding additional feature extractor in the models to improve local feature recognition.

5.8 Summary

A novel recursive tool path prediction framework has been proposed for the forming sequence planning of a consecutive rubber-tool forming process. A deep learning model is the major component in the framework, which predicts the optimal tool path by learning the topological relationship between the current and target workpiece shape. Three series of deep learning models have been implemented, of which the single feature extractor and cascaded network can learn from the spatial feature in input images while the LSTM can learn both spatial features and temporal characteristics from the inputs.

A single feature extractor is not able to predict the optimal tool path for the consecutive rubber-tool forming process, while it has nontrivially better performance in learning the forming parameters for a single deformation, with about 15% improvement in generalisation compared to that trained with data including multiple deformations.

With Dataset within 2 punches, the cascaded network, which includes an extra image reconstruction sub-model, outperforms the other two series of models. In addition, cGAN is found to precede all other sub-models in the accuracy and consistency of reconstructing the target image. The cascaded network with cGAN and the pre-trained VGG16 has the most superiority among models trained with Dataset within 2 punches.

With Dataset within 3 punches, both of the single feature extractor and the cascaded networks and can hardly learn the true function of tool path prediction and fail to predict the optimal forming parameters. However, both single and stacked LSTM models exhibit promising prediction results with the generalisation accuracy over 90%, which blatantly precedes the best model trained with Dataset within 2 punches. In addition, the single LSTM marginally outweighs the stacked LSTM by 1% - 2% in this study.

The performance of the LSTM models is improved by over 15% when more training data are provided, even though the additional training data do not enrich the data diversity of the dataset but only include forming results from 3 punches in the same forming pattern. Besides, with additional data, the LSTMs manage to learn the optimal tool path of a more complex deformation process. However, as the learning complexity increases, the performance of both the single feature extractor and cascaded networks blatantly deteriorates by over 10%.

High ratio of mispredictions from the LSTM models is found on location 101 and stroke 1 mm, which could be improved by dataset structure design and incorporating additional local feature extractors.

Chapter 6

Unlimited punches: a generalisable tool path planning strategy

6.1 Introduction

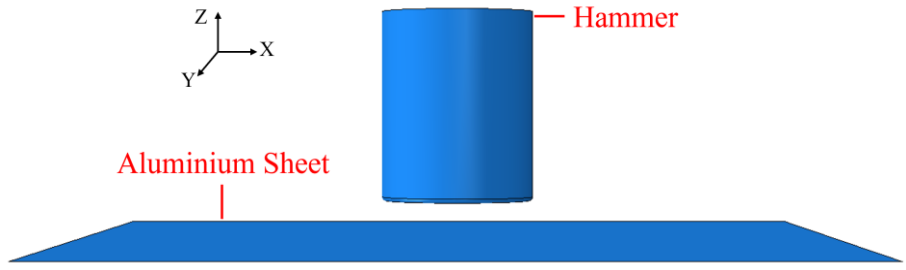
In addition to using DNNs for tool path learning and generation from training dataset, reinforcement learning algorithms can also be applied to learn the optimal tool path from scratch, which can be used to configure the Process Learning Module in the intelligent sheet metal forming system as shown in Figure 3-2. However, unlike numerous supervised learning applications in metal forming industry, reinforcement learning has been significantly ignored due to the expensive data acquisition.

This chapter presents a preliminary study of utilising reinforcement learning technology for tool path planning, based on which a generalisable tool path planning strategy is proposed. Deep Q-learning algorithm was implemented to learn a single deformation step, including a sequence of tool movement followed by a punching operation. Section 6.2 presents the preliminary study, including process modelling, RL configuration, learning results and results discussion, which has been presented at 18th International Conference Metal Forming 2020 [170]. Section 6.3 presents a novel generalisable tool path planning strategy, which utilises both deep reinforcement learning and supervised learning technologies.

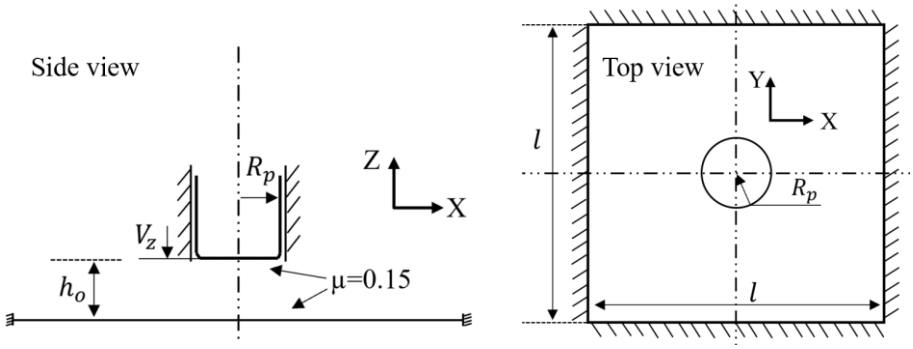
6.2 A preliminary study on reinforcement learning application to one single deformation step

6.2.1 FE modelling

Figure 6-1 shows the 3-D model and FE model of a hammering process. As shown in Figure 6-1a, a hammer was placed above an AA1050 sheet at a fixed distance. The hammer was allowed to translate along X and Y axis at a constant feed rate and drop at a constant speed along Z axis to deform the aluminium sheet. During each punch, the hammer dropped at a chosen speed for 0.2 s and lifted up at 400 mm/s for 0.05 s. The FE models of the hammer and sheet metal are shown in Figure 6-1b, with dimensions detailed in Table 6-1. To evaluate the performance of the deep Q-learning algorithm in searching the optimal tool path, a target forming shape of this case was set to be the shape deformed by a single punch at location (x, y) = (-10, -5) with punching speed of 100 mm/s.



(a) 3-D CAD model



(b) FE model

Figure 6-1. The 3-D model and FE model of the hammering process.

Table 6-1. Test setup details for hammering modelling (unit: mm).

Features	Workpiece length, l	Punch radius, R_P	Original height, h_o	Workpiece thickness, t
Value (mm)	200	20	10	1

To collect data for training the RL model, Abaqus 2019 was used to perform computational experiments. The material and mesh properties are presented in Table 6-2, with the total amount of mesh determined in Section 6.2.3.1.

Table 6-2. FE modelling details for the hammering process.

Component	Young's modulus, E	Poisson's ratio, γ_p	Mesh property
Sheet metal	71 GPa [171]	0.33	Deformable shell, homogeneous
Hammer	-	-	Rigid shell

6.2.2 Deep reinforcement learning setup

Deep Q-learning was used in this study for tool path learning, which has been defined in Eq. (2-18). The transition of this RL model was designed as follow:

- State (s_t): it included the hammer position coordinates, the sheet metal mesh coordinates and the Von-Mises stress distribution. It was pre-processed and input to the DQN.
- Reward (r_t): if the shape of next state (s_{t+1}) was the same as the desired shape (within a tolerance of ± 0.01 mm), $r_t = Q(s_t, a_t) = 100$; if not, $r_t = -1$.
- Action (a_t): the actions are listed in Table 6-3, in which x and y represent hammer translation direction and "*Vel.*" is its punching velocity.
- Next state (s_{t+1}): it is the state after taking action (a_t) at state (s_t), having the same type of data with state (s_t).

Table 6-3. Prescribed actions for the hammer (unit for x/y: mm; unit for Vel: mm/s)

Action number	1	2	3	4	5	6	7	8	9	10
Action type	*x*	*x*	*y*	*y*	*Vel.*	*Vel.*	*Vel.*	*Vel.*	*Vel.*	*Vel.*
Value	+5	-5	+5	-5	50	100	150	200	250	300

A CNN with 6 convolutional layers (valid padding) and 3 fully connected layers, as shown in Figure 6-2, was used as DQN to approximate the action-value function. The neural network was established and trained using TensorFlow Core v2.2.0, with 178,482 trainable parameters in which the total number of weights was 178,130. In the network configuration, the input was the state s_t exported from Abaqus simulations. The input data was pre-processed to a 116 × 24 array and then fed into the DQN. The outputs were the predicted Q-value for each prescribed action shown in Table 6-3.

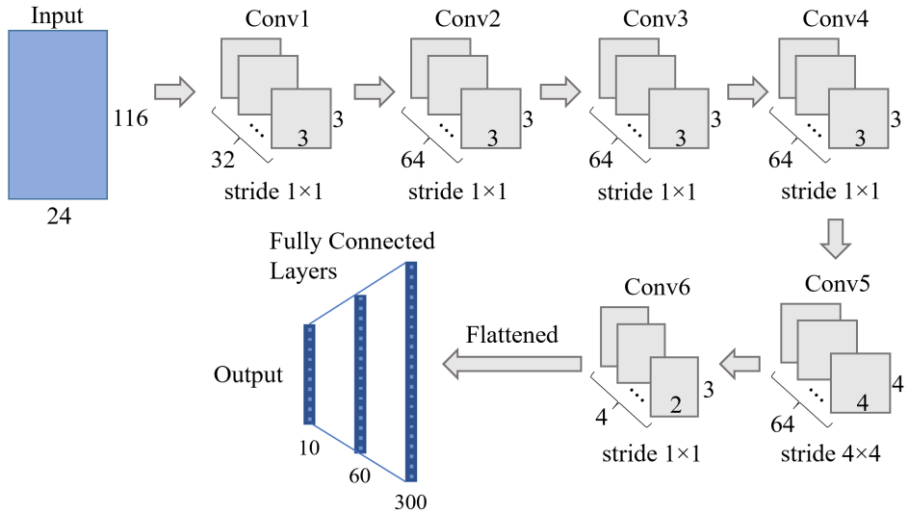


Figure 6-2. DQN architecture for the RL process.

A python-based overall learning workflow was developed based on the deep Q-learning algorithm, and its learning loop is schematically illustrated in Figure 6-3.

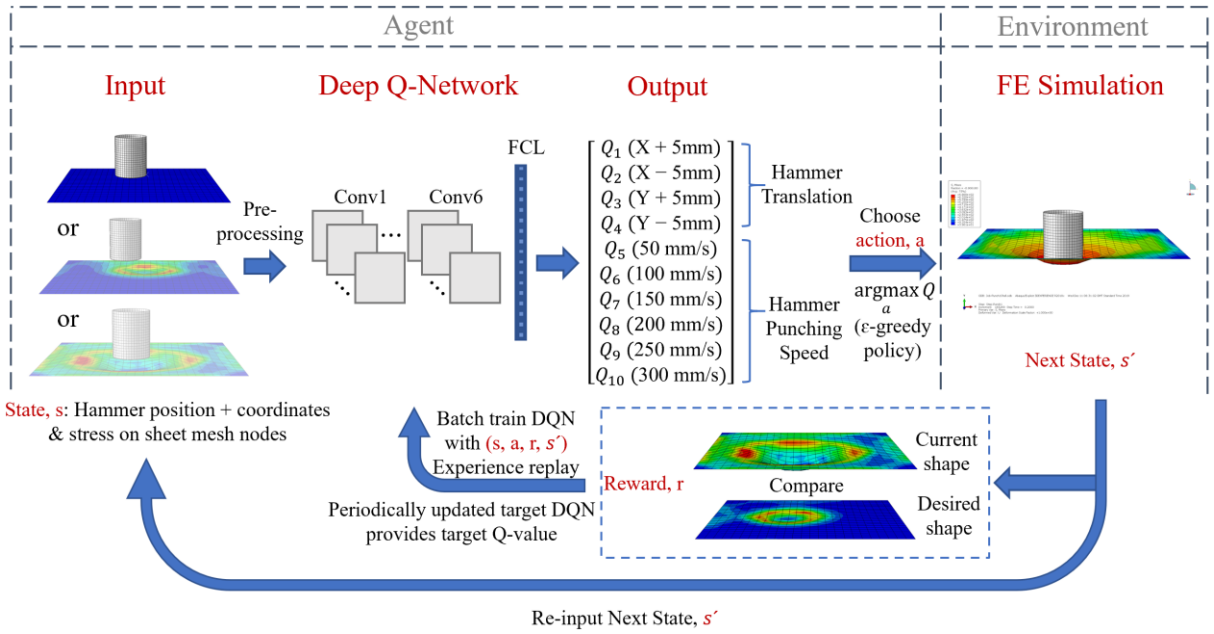


Figure 6-3. Overall learning architecture embedded with deep Q-learning.

In the learning process, the hammer would be reset to its initial location (x, y) = (0, 0) if a learning loop, termed episode, terminated. The episode would terminate either an action of punch was taken by the agent or the goal shape was achieved. The overall learning process can be briefly summarised as follow:

1. Hammer is reset to its initial location (x, y) = (0, 0).

2. Pre-process the input and feed it into the DQN to get the Q-value for each action, choose the action based on ε-greedy policy [95] with ε decaying from 1 to 0.1 in 20000 training steps. ε then remains constant at 0.1 to keep the chance of exploration of the learning agent in case that optimal forming solution has not been found yet.
3. Perform the simulation based on the action selected from step 2 and obtain the new sheet shape.
4. Compare the new sheet shape and the desired shape. If the absolute error is within 0.01 mm, let the term $r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \theta^-) = 100$ in Eq. (2-18); otherwise, $r_t = -1$. Store the new transition in the experience replay buffer and sample 64 transitions (minibatch size) from the buffer. Compute the loss according to Eq. (2-18) for the batch and train the DQN with Root Mean Square Propagation (RMSProp) optimiser [59], learning rate of 2.5e-5 and discount factor of 0.9.
5. If any punching action is selected, repeat the process from step 1; otherwise, Set the new shape from step 3 as the next state and re-input it to the DQN after pre-processing.
6. Repeat step 1-5 until the DQN training is converged.

6.2.3 Learning results

6.2.3.1 Grid-independence study

Grid-independence study was performed for this study to determine a proper selection of mesh size which was desirable in terms of both computation accuracy and computational time. The simulations for grid-independence study were performed under the same boundary conditions as introduced in Section 6.2.1, in which the hammer was at its initial position and punching the sheet-metal with speed of 100 mm/s. Five simulations were created with different mesh size on the sheet metal part, the maximum stress in each simulation was compared as shown in Table 6-4.

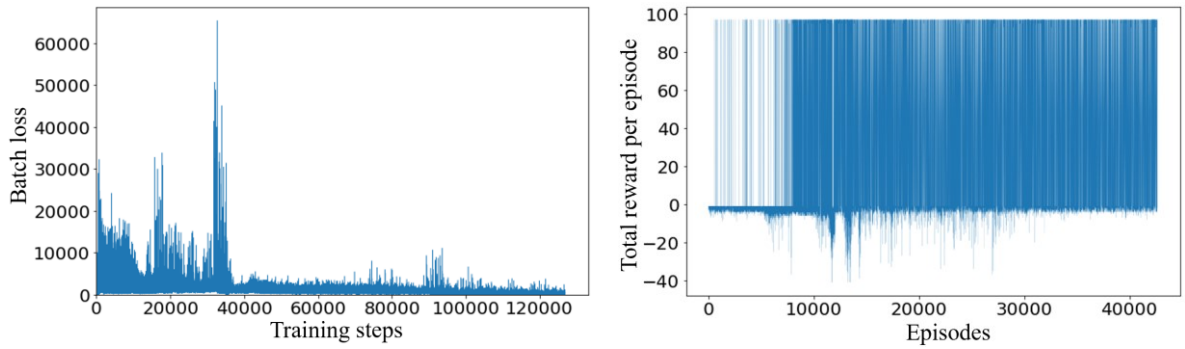
Table 6-4. Grid-independence study results.

	Case 1	Case 2	Case 3	Case 4	Case 5
Mesh size on sheet part (mm)	8	6	4	3	2
Total mesh nodes on sheet part	676	1156	2601	4624	10201
Max. Stress (MPa)	72.69	72.97	75.20	75.97	78.06
Computational time (min)	2	4	14	36	118

As presented, max stress occurring in each simulation case increased, in general, as the mesh on sheet-metal part became denser. The case with the most refined mesh yielded higher maximum stress than that with the coarsest mesh by approximately 7.39%, which, however, can be traded for about 98.3% reduction in computational expense. Also, the RL training process requires significant amount of data which necessitates low computational time for each simulation. Thus, to guarantee low simulation cost and acceptable computational accuracy, case 1 with mesh size of 8 mm on sheet-metal part was selected as the "reference case" to provide computational data for RL training.

6.2.3.2 RL training results

With the overall learning workflow shown in Figure 6-3, the DQN was trained to convergence. Figure 6-4 shows the training history of batch loss and total reward in each episode regarding to training steps and number of episodes, respectively. It can be seen the loss decreased to a low value in the end of the training. The total reward history showed that the target shape was achieved only a few times before about 8000 episodes, which was indicated by the sparse distribution of high total rewards at the start of training. As the learning continued, the frequency of meeting the goal significantly increased. However, some troughs of total reward occurred until about 32000 episodes, which indicated that the DQN predicted incorrect forming route thus accumulating negative rewards during this period. The total reward troughs disappeared in the last 10000 episodes, marking the convergence of the RL model. As the ε decreased to 0.1 rather than 0, the learning agent would choose random actions at times. Thus, the total reward alternated between positive and negative values after convergence.



(a) History of batch loss **(b) History of total reward per episode**

Figure 6-4. RL training process in terms of batch loss and total reward per episode.

Figure 6-5 shows the history of Q-values for three actions for hammer at initial position and position of (x, y) = (0, -5). For brevity, only the Q-values for three important actions

(punching with velocity of 100 mm/s and translating the hammer in -X or -Y direction) are depicted. The Q-values for other actions have also been checked and would not influence elaboration. At the initial position, the Q-values for the two translation actions converged to the same level, which were higher than that for the punching action. This indicated that translation actions would be selected at initial position with nearly equal probability. At position of (x, y) = (0, -5), the action of translating the hammer in -X direction was selected as its Q-value was the highest after convergence. It can be observed that there was a surge just before the 40000 training step, which was due to the high training loss shown in Figure 6-4a. The unconverged model predicted wrong forming routes, thus leading the agent to other state spaces and inducing more exploration if the state space had not been searched yet. The exploration was then found unnecessary for meeting the goal, the instantly increased Q-value was thus recovered afterwards.

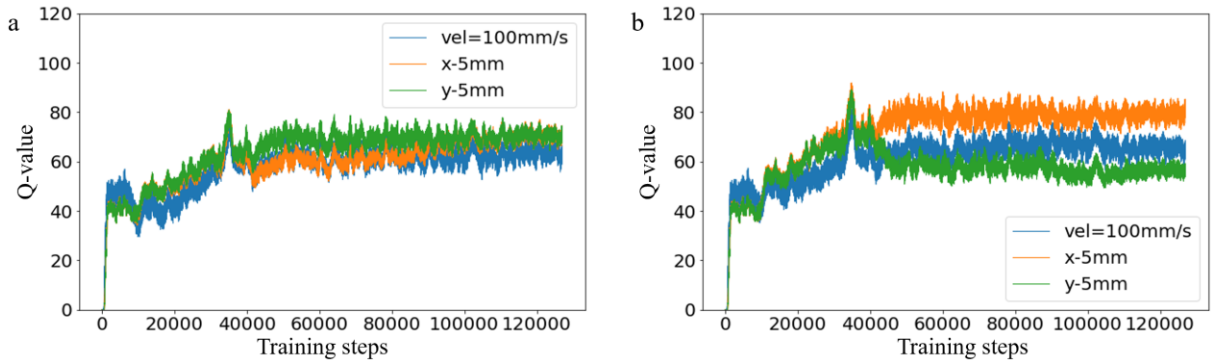


Figure 6-5. Q-values versus training steps for hammer at (a) initial position (x, y) = (0, 0); (b) (x, y) = (0, -5).

Figure 6-6 shows the history of Q-values for the three actions for hammer at positions of (x, y) = (-5, 0) and (x, y) = (-5, -5). At position of (x, y) = (-5, 0), the action of translating the hammer in -Y direction was selected after convergence. At position of (x, y) = (-5, -5), the action of translating the hammer in -X direction was selected.

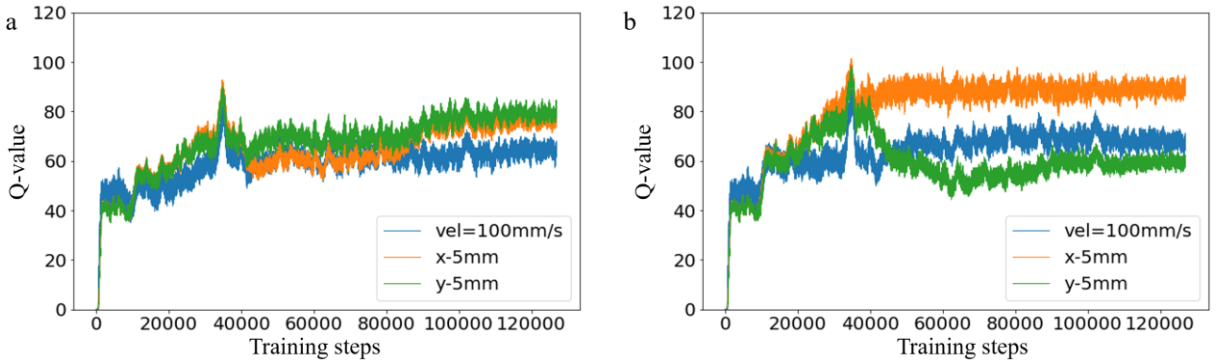


Figure 6-6. Q-values versus training steps for hammer at (a) (x, y) = (-5, 0); (b) (x, y) = (-5, -5).

Figure 6-7 shows the history of Q-values for three actions for hammer at position of (x, y) = (-10, -5). At this location, the action of punching with velocity of 100 mm/s was selected. The Q-value for the action of punching velocity of 100 mm/s fluctuated about 100 after convergence, which agreed with the designed reward scheme.

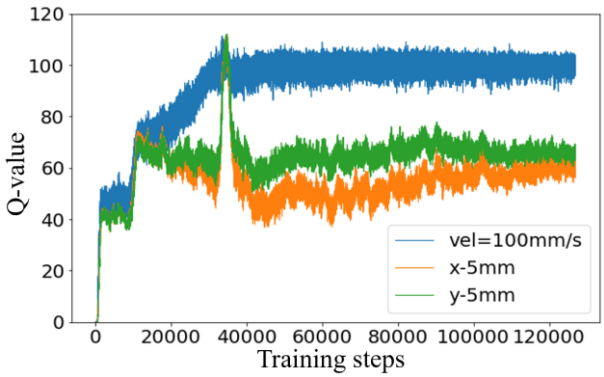


Figure 6-7. Q-values versus training steps for hammer at (x, y) = (-10, -5).

6.2.4 Discussion

The successful training of the RL model with the developed overall learning workflow in this section indicates the possibility of intellectualising the traditional free-form sheet-metal stamping process, by harnessing the state-of-the-art reinforcement learning technology. With no expertise provided in prior, the RL model is able to predict the optimal forming route through learning from the trial-and-error process. Consequently, obstacles impeding the development of the traditional free-form sheet-metal stamping, such as its high reliance on individual experience and low repeatability, can be overcome.

However, a common issue hindering applications of reinforcement learning in engineering problems is that the FE computation for each forming operation can be prohibitively expensive. Besides, generalisation gap is a common issue in RL applications [172], which is a challenge under constant research. An evident reason leading to its inferior generalisation is that the data collected during RL training are mostly lying on the path towards a certain optimisation target. With a completely different target, the model would fail in generalisation since it was trained without useful data towards the new target.

6.3 Generalisable tool path planning strategy

To improve the generalisation of tool path planning of consecutive free-form sheet metal stamping process, a generalisable tool path planning strategy has been proposed. This strategy works by segmenting the target workpiece, based on the shape of three groups of segments classified in prior, into a few segments whose subpaths are generated through deep learning approach. The entire tool path for the target workpiece would be acquired by aggregating the subpaths for all workpiece segments. By classifying common groups of segments with the same shape features, any arbitrary workpiece can be regarded as assembled by segments from these groups. Through dynamic programming, the tool path learning complexity for a complete workpiece was reduced to simpler subproblems of path learning for each group of workpiece segments. As the segments in each group are highly correlated in shape, the tool path learning for each segment group is significantly more generalisable than that for arbitrary workpieces.

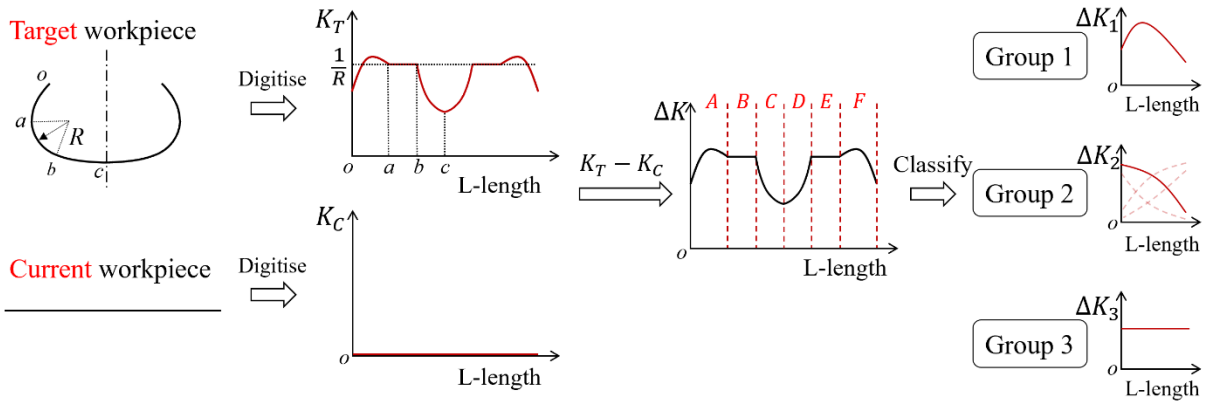
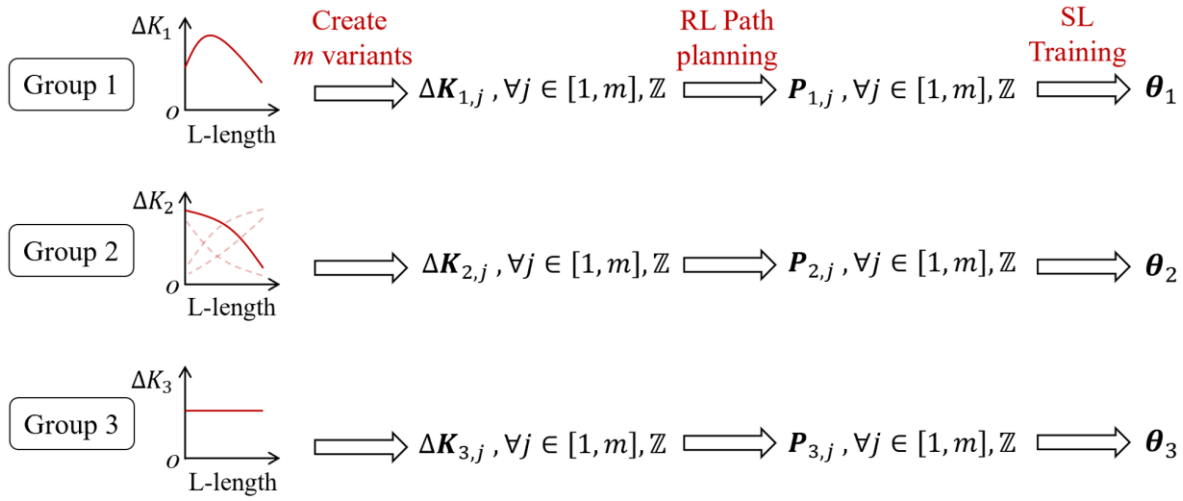


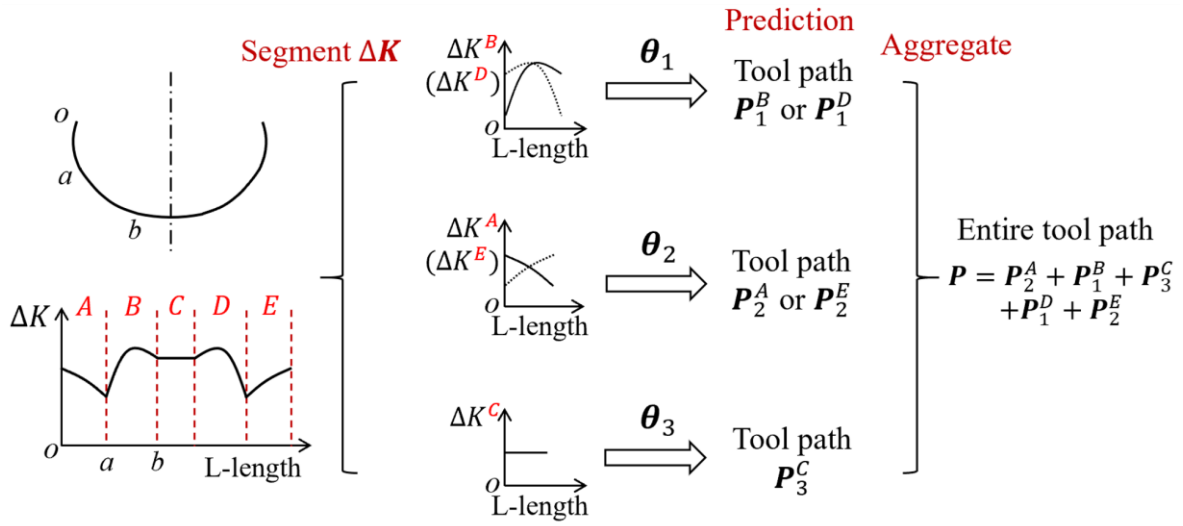
Figure 6-8. Schematic digitisation procedure for workpiece state representation ΔK-graph and the classification of three groups of segments. The drawings for target and current workpieces depict their top surfaces. The dashed lines in Group 2 signify other segments having the same shape features as the solid line, which are also counted in this group. L-length denotes longitudinal length.

To quantitatively measure the shape difference between the target and current workpiece, a curvature difference distribution graph ($\Delta\boldsymbol{K}$-graph) was generated by subtracting the current $\boldsymbol{K}_C$-graph from the target $\boldsymbol{K}_T$-graph to represent the workpiece state, as shown in Figure 6-8. The current workpiece was considered to be close to its target shape if the value of $\Delta\boldsymbol{K}$ approached zero at any point along the longitudinal length. From the example in Figure 6-8, the $\Delta\boldsymbol{K}$-graph was split into 6 segments, A-F. Through the segmental analysis of the $\Delta\boldsymbol{K}$-graphs of various workpieces, three groups of segments were classified, of which any arbitrary $\Delta\boldsymbol{K}$-graph can be composed. Groups 1 and 2 consisted of half-wave shaped and quarter-wave shaped segments, and Group 3 included constant-value segments representing circular arcs or flat sheet.

There were two phases in the generalisable tool path planning strategy, learning phase and application phase, as shown in Figure 6-9. At the learning phase, for each group of segments, m variants of $\Delta \boldsymbol{K}$-graphs, $\Delta \boldsymbol{K}_{i,j}$, were created as shown in Figure 6-9a, where i is the group number and j is the variant number. The tool path, $\boldsymbol{P}_{i,j}$, for each of the variant of segment in each group was then learned and planned through deep reinforcement learning, without any path planning experience in prior. After the tool paths for all segments were obtained, a deep supervised learning model was trained with the tool path data, $\boldsymbol{P}_{i,j}$, for each group to generalise the efficient tool path patterns for segments from each group.



(a) Learning phase: learning process for three groups of workpiece segment. The subscript *i,j* denote the *i*th group and *j*th variant of segment.



(b) Application phase: path prediction process for a new workpiece. The subscript denotes the group number and the superscript denotes the segment number from the ΔK-graph.

Figure 6-9. The generalisable tool path planning strategy through deep reinforcement and supervised learning.

At the application phase, as shown in Figure 6-9b, a new workpiece was firstly digitised to the $\Delta \boldsymbol{K}$-graph and segmented in accordance with the three groups. Five segments, A-E, were obtained in this example, and their tool paths were predicted using the deep supervised learning model trained for their particular groups at the learning phase, respectively. At last, the entire tool path for the workpiece was obtained by aggregating all the tool path for each segment.

6.4 Summary

In this chapter, a reinforcement learning algorithm, deep Q-learning, has been attempted for tool path planning of a simplified free-form sheet metal stamping operation. The DQN model can successfully learn the optimal tool path for this operation, which indicates the potential of harnessing the state-of-the-art reinforcement learning techniques to automate and intellectualise the traditional smithing operations. Due to the low generalisation of pure reinforcement learning approach, a generalisable strategy has been proposed for tool path planning of arbitrary sheet metal parts. This strategy factorises target workpiece into segments and optimises tool path through dynamic programming, utilising deep reinforcement learning and supervised learning technologies.

Chapter 7

Instantiation of the generalisable tool path planning strategy

7.1 Introduction

To verify the utility of the generalisable tool path planning strategy proposed in Section 6.3, this chapter presents the instantiation of this strategy. Section 7.2 presents the experimental setup for the consecutive forming process, followed by its digitisation method. The consecutive forming characteristics are studied and presented in Section 7.3, from which the forming parameters and forming goal are designed. Section 7.4 presents the setup of six deep reinforcement learning algorithms and a virtual environment for algorithms investigation. The most superior DNN for tool path planning from Chapter 5 is extended in Section 7.5, with a description of training methods. Section 7.6 presents the learning results including the selection and investigation of RL algorithms as well as the learning generalisation performance of different DNNs. Finally, the generalisable strategy is verified through a case study in Section 7.7.

7.2 Experiments test setup of consecutive forming process and forming digitisation

The rubber-tool forming process proposed in Section 5.2 was adopted to consecutively deform a sheet metal while retaining a sound surface condition during the forming process. From the test setup and FE model shown in Figure 7-1, the workpiece was deformed by a rubber-wrapped punch on a workbench rubber. The specification of the setup is summarised in Table 7-1. The deformation process was set up, for simplicity, in 2-D space and computationally

performed with Abaqus 2019. The FE plain strain model was configured with the material of AA6082 for the workpiece and natural rubber for the punch rubber and workbench rubber as in Section 5.2. Mesh for the workpiece was of size 0.1 mm, with 17,164 elements in total.

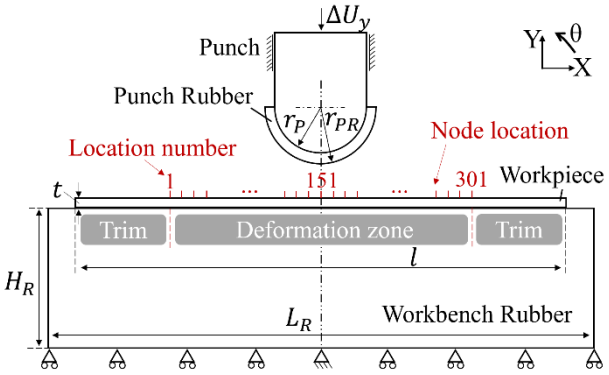


Figure 7-1. Test setup and FE model for the rubber-tool forming. The lengths of the deformation and trim zones are 30 mm and 10 mm, respectively.

Table 7-1. Test setup details for the rubber-tool forming.

Features	Workpiece length, l	Workpiece thickness, t	Punch radius, r_P	Punch rubber radius, r_{PR}	Workbench Rubber length, L_R	Workbench Rubber height, H_R
Value (mm)	50	1	5	6	58	56

To realise the consecutive forming process in FE simulations, the forming process was digitised and standardised for precise process control. As shown in Figure 7-1, the workpiece was divided into two zones, namely deformation zone and trim zone, with lengths of 30 mm and 10 mm, respectively. The punch could only work on the deformation zone, and the trim zone would be trimmed after the deformation had completed. The trim zone was reserved without deformation due to that the significant shear force from the edge of the workpiece could easily penetrate the workbench rubber, which would cause non-convergence issue in FE computation. The deformation zone was marked by 301 node locations, numbered from left to right, which were consistent with the mesh node locations.

To quantitatively observe and analyse the workpiece shape, a curvature distribution (***K***) graph was generated to represent the shape of the workpiece deformation zone, as shown in Figure 7-2. The local curvature K of a point on the workpiece was calculated by *Menger* curvature, which is the reciprocal of the radius of the circle passing through this point and its two adjacent points. Thus, a total of 303 mesh nodes on the top surface of the workpiece were used to generate the ***K***-graph, including 301 nodes in the deformation zone and one additional

node next to each end of the deformation zone. Using 0.1 mm of interval distance between each two contiguous node locations, the workpiece shape can be regenerated from its $\boldsymbol{K}$-graph.

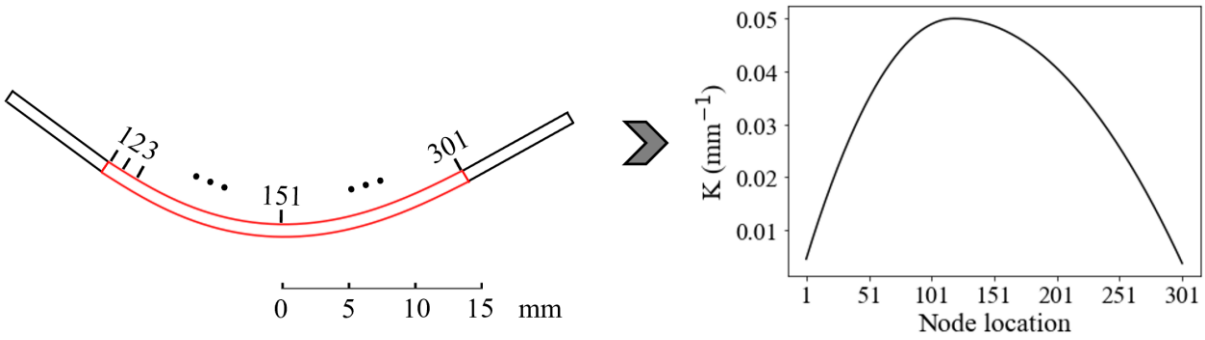


Figure 7-2. Example of workpiece shape (left) and its curvature K distribution along node locations (right). The region highlighted in red in the drawing denotes the deformation zone, and the K-graph is generated from the workpiece top surface of this zone.

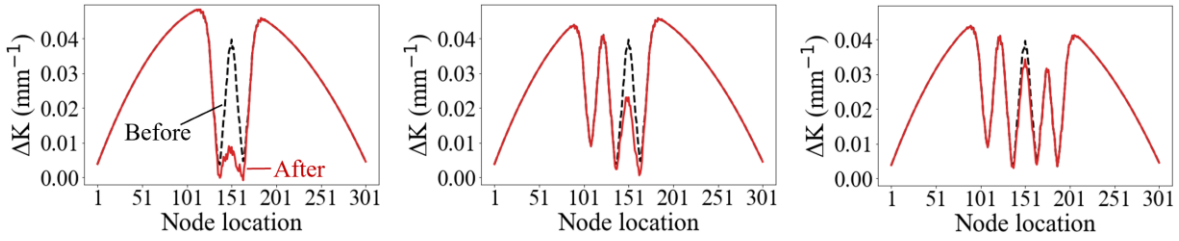
7.3 Forming goal and forming parameters design

In the context of the free-form sheet metal stamping test setup presented in Figure 7-1, at each step of the forming process, the stamping outcome is determined by the punch location and punch stroke. However, the large amount of punch location options, 301 in total, would incur considerably vast search space for the tool path planning problem. Thus, to simplify the problem, a forming heuristic (Heuristic 1) was applied to this forming process, which is in conformity with practical forming scenario, to allow the node location that had the most salient shape difference from the target workpiece to be selected at each forming step. In a word, the node location where the value of ΔK was highest in the $\Delta\boldsymbol{K}$-graph was selected at each step.

As the workpiece shape is close to its target when the ΔK approaches zero at any point along the longitudinal length, the goal of the consecutive forming in this research was considered to be achieved if $\max(|\Delta\boldsymbol{K}|) \leq 0.01$ mm^{-1}. Thus, in order to determine an appropriate range of punch stroke values to select during deformation, by which the forming goal is possible to achieve within a relatively small search space, a preliminary study was performed to investigate the consecutive forming characteristics. Two phenomena, namely close punch effect 1 and 2 (CPE1 and CPE2), were discovered in this study, which are shown in Figure 7-3 and Figure 7-4.

From Figure 7-3, the $\Delta\boldsymbol{K}$-graphs of three workpieces before and after the same punch with stroke of 3.0 mm at the location of 151 are shown. The three workpieces had been consecutively

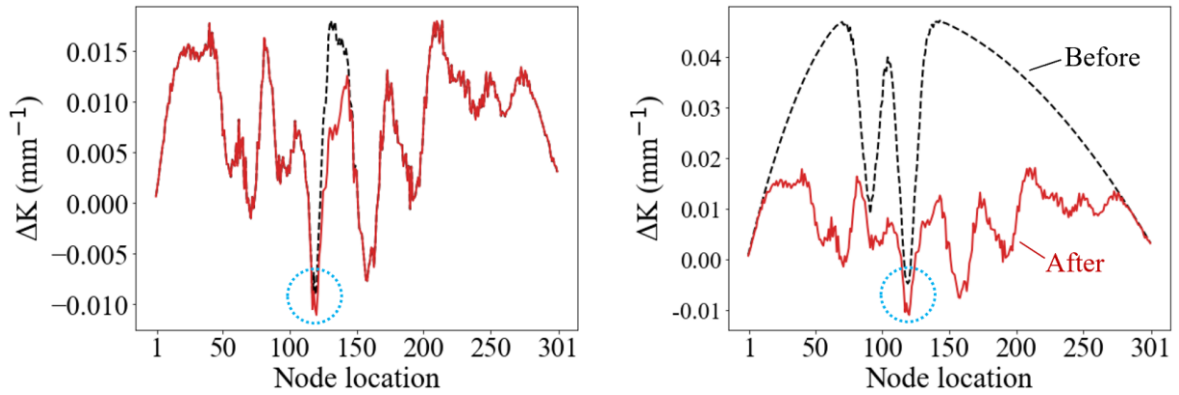
deformed by 2, 3 and 4 punches in the vicinity of this node location, as shown respectively in Figures 7-3a, 7-3b and 7-3c. It can be seen that more prior deformation underwent near the node location of interest, less deformation was resulted in, i.e., larger punch stroke was required to accomplish a certain change of shape at this location. This phenomenon was named CPE1, which barely escalated with more than 4 punches in prior.



(a) 2 punches in prior **(b) 3 punches in prior** **(c) 4 punches in prior**

Figure 7-3. Close punch effect 1 on the punch with stroke of 3.0 mm at the location of 151. (a), (b) and (c) present the ΔK-graphs before and after this punch on the workpiece which has been consecutively deformed, in prior, by 2, 3 and 4 punches, respectively.

Figure 7-4 shows the $\Delta \boldsymbol{K}$-graphs of two workpieces before and after 1 punch and 50 punches, respectively.



(a) 1 step difference after punch at location of 132 **(b) 50 steps difference**

Figure 7-4. Close punch effect 2 from the punches near location of 118. (a) and (b) present the ΔK-graphs before and after 1 punch and 50 punches, respectively. The area highlighted by dashed circle is where CPE2 was found.

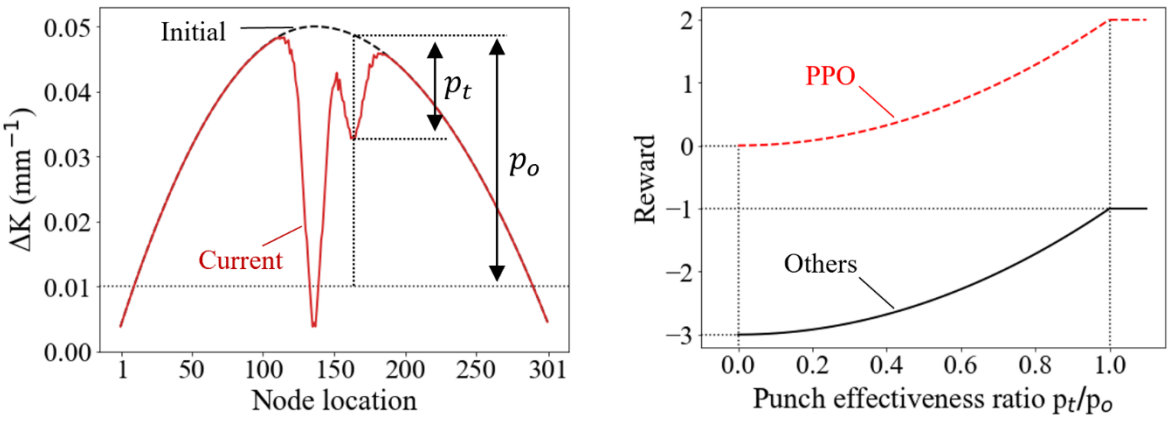
From Figure 7-4a, it can be seen that the ΔK values around node location of 118 was decreased by about 0.002 mm^{-1} after deformation applied to location of 132, although the punch at the latter location had less effect on the ΔK value than that at the former location did. From Figure 7-4b, it can be seen that the workpiece had been deformed at location of 118 since the 2^{nd} step, and the ΔK value at this location was affected by the punches nearby in the following

50 steps and decreased by about 0.008 mm^{-1}. This phenomenon was named CPE2, whose area of influence covered approximately 5 mm (about 50 node locations) around the node location.

From the analysis above, 19 options of punch strokes were determined, ranging from 2.1 mm to 3.9 mm in 0.1mm increments.

7.4 Deep reinforcement learning setup and virtual environment

7.4.1 Learning setup and hyperparameters



(a) Schematic for reward evaluation **(b) Reward function**

Figure 7-5. Reward function and its evaluation for each action. (a) shows the reward evaluation method for punch stroke of 2.2 mm at location of 162, and the two lines denote the initial and current ΔK-graph; (b) shows the reward function for evaluation.

For RL environment, Abaqus 2019 was interfaced with the RL algorithms to supply computation results during learning. The transient (s_t, a_t, r_t, s_{t+1}) was formulated as follow:

- The state/next state was in the form of workpiece state representation $\Delta \boldsymbol{K}$-graph and a one-hot vector of size 1 × 301 indicating the punch location.
- The action was the punch stroke, ranging from 2.1 mm to 3.9 mm (19 in total for discrete action space).
- The reward was defined to measure the goodness of the selected action at given state, whose evaluation is shown in Figure 7-5. After each action at a given state, the reward was determined by the punch effectiveness ratio, which was defined as the ratio of the punch effect on ΔK at given location at time step t (p_t) to the expected effect at this location (p_o), with the function $r_t = 2(p_t/p_o)^2 - 3$ (except for PPO: $r_t = 2(p_t/p_o)^2$). p_o is the distance between the threshold and the point on initial $\Delta \boldsymbol{K}$-graph. If the workpiece was overpunched,

the ΔK at the punch location below the lower threshold $-0.01\ mm^{-1}$, $r_t = -100$ (PPO: $r_t = -1$ and DDPG: $r_t = -3$); if the forming goal was achieved, i.e. $\max(|\Delta \boldsymbol{K}|) \leq 0.01\ mm^{-1}$, $r_t = 0$ (PPO: $r_t = 500 - 2.5 \times$episode step). Negative rewards were used for each step to penalise unnecessary steps, except for PPO where unnecessary steps were penalised by rewarding early termination. A reward of -3 was assigned for overpunch in DDPG learning rather than -100 since it was found that sparse rewards can cause failures in DDPG training [173].

Figure 7-6 presents the reinforcement learning process configured for the tool path learning purpose, using FE simulations as the RL environment. The RL process collected data in a loop, starting from digitising the workpiece geometry to the state s_t and feeding it to the learning agent. The learning agent predicted the stroke a_t based on the current policy and exploration scheme. The FE simulation was configured by repositioning the current workpiece about the punch location and setting up the selected punch stroke, and the deformed workpiece geometry was extracted and stored. The deformed geometry was also digitised to obtain the next state s_{t+1}, with which the reward r_t was evaluated through the reward function. The collected transient (s_t, a_t, r_t, s_{t+1}) at this time step was then used to optimise the objective function J and update the agent policy. The RL loop repeated by re-inputting the next state to the agent as the state in the next loop.

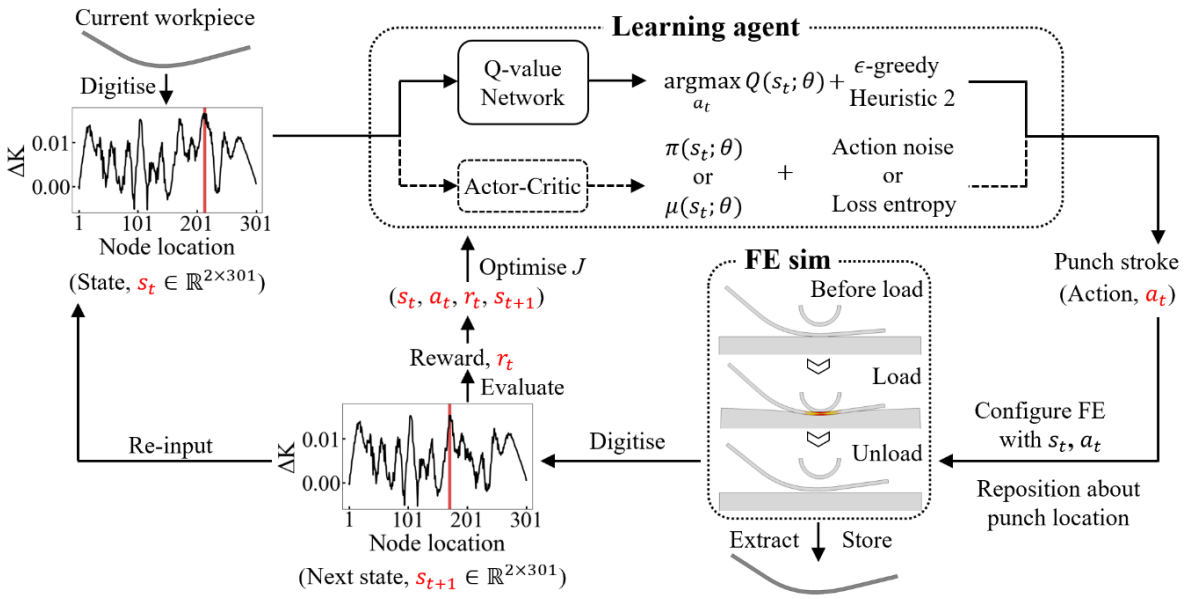


Figure 7-6. The reinforcement learning process for the tool path learning of the consecutive rubber-tool forming process, of which the FE simulation (FE sim) was used as the RL environment to provide real-time deformation results. The vertical line in the ΔK-graph denotes the punch location.

The learning methods for the six RL algorithms, which are all model-free algorithms, are shown in Table 7-2. The off-policy algorithms were trained with experience replay, of which all learning histories were stored to be uniformly sampled in minibatch for training, while the on-policy algorithms were trained with the immediate experience. Target network was used for action evaluation, which was updated by the online network periodically for stable learning progress. For exploration and exploitation, the Q-learning algorithms adopted ϵ-greedy policy, while A2C used an additional entropy term from [174] in the loss function and DDPG used a *Gaussian* distributed action noise. In addition to above, a forming heuristic (Heuristic 2) was developed to facilitate the learning process, which was defined as follows: the choice of the stroke at the current node location, if applicable, cannot be less than previous choices at the same location in one run, otherwise a larger value of stroke was randomly selected for this location. This heuristic was only applied in addition to ϵ-greedy policy as they have the same exploration mode, which would not disturb the training data structure.

Table 7-2. Learning methods for the six reinforcement learning algorithms (Y – True, N – False).

Algorithm	On/Off-policy	Target network	Explore/Exploit	Heuristic 2
DQN	Off-policy	Y	ϵ-greedy	Y
Double-DQN	Off-policy	Y	ϵ-greedy	Y
Dueling-DQN	Off-policy	Y	ϵ-greedy	Y
A2C	On-policy	N	Loss entropy	N
DDPG	Off-policy	Y	Action noise	N
PPO	On-policy	Y (old policy)	-	N

The learning hyperparameters for RL are summarised in Table 7-3. The maximum step per episode signifies the maximum forming step allowed for each run of the consecutive forming trial. The episode would end if any of the following conditions was met: 1) forming goal achieved, 2) overpunch and 3) maximum step per episode (step/ep) attained. It is noted that the target network for Q-learning was updated every 20 learning steps, while that for DDPG was softly updated with $\tau = 0.01$ and that for PPO was updated every *rollout* (512 steps in this research) of the online policy. For ϵ-greedy policy, the value of ϵ decayed from 1.0 to 0.1.

Table 7-3. The learning hyperparameters for training the six RL algorithms. step/ep denotes step per episode and freq. denotes frequency.

Hyperparameter	Value	Hyperparameter	Value
Learning rate	$[10^{-4}, 10^{-1}]$	Discounting ratio γ	0.9
Batch size	16	Target net update freq.	Every 20 step
Maximum step/ep	100	ϵ decay step (ϵ-greedy)	400

With regard to the models used for value function and policy function approximations in all six algorithms, the learning performance of a shallow multilayer perceptron (MLP) and a convolutional neural network (CNN) were compared as in [100]. ReLU was used for all hidden layers. There was no activation function for the output layer of the value network, while *softmax* was used for that of all policy networks (except for DDPG where *tanh* was used). The MLP had 2 hidden layers with 400 and 200 units, respectively (164,819 parameters). The network had 2 inputs, the $\Delta \boldsymbol{K}$-graph and the one-hot vector for punch location, each followed by a half layer of neurons in the 1st layer before they were added together and fed into the 2nd layer. The CNN had the same architecture with the one used in [96], with an additional hidden layer with 512 units, for the 2nd input (punch location), whose output was added to the last hidden layer of the original CNN (1,299,891 parameters).

7.4.2 Virtual environment for RL algorithms comparison

Subject to the FE computational speed, it is considerably time-consumptive to test the feasibility of tool path learning for the consecutive forming process using RL. Thus, a virtual environment was developed to imitate the rubber-tool forming behaviour by having similar punch effects on the $\Delta \boldsymbol{K}$-graph to those computed by FE simulations, with which the performances of the six RL algorithms in tool path learning were compared. The virtual environment was composed to also manifest CPE1 and CPE2 as presented in Section 7.3. The effect of stroke value on the $\Delta \boldsymbol{K}$-graph was also imitated by the virtual environment, which was defined through a parametric study. The details of configuring the virtual environment are presented in Appendix A, and the imitation fidelity of the virtual environment will be discussed in the end of Section 7.6.1.

7.5 Deep supervised learning models and training methods

From the introduction of generalisable strategy in Section 6.3, 3 groups of workpiece segments were classified. In the segmental analysis of the $\Delta \boldsymbol{K}$-graphs, the segments from Group 1 were found to appear most frequently among those from three groups. Thus, Group 1 was used for instantiation of the generalisable tool path planning strategy, and a total of 25 variants of segments in this group were arbitrarily created through the method in Appendix B.

After the optimal tool paths for the 25 variants of workpiece segments in Group 1 were acquired from deep reinforcement learning, they were used to train deep supervised learning models to learn the efficient tool path patterns for this group.

7.5.1 Deep neural networks

Three deep neural networks (DNNs), namely single CNN, cascaded networks and CNN LSTM, had been compared in predicting the tool path through a recursive prediction framework in Chapter 5. Since the results revealed that the performance of CNN LSTM preceded that of the other two models, CNN LSTM was adopted in this research, with VGG16 [89], ResNet34 and ResNet50 [91] as the feature extractor, respectively. The model architectures for these models were the same as those used in Section 5.5.3, with a simple substitution of feature extractor with ResNet34 and ResNet50. The input to the LSTM was the partial forming sequence made up of the concatenation of the $\Delta\boldsymbol{K}$-graph and the punch location vector for each time step, and the output from the model was the punch stroke prediction for the coming step. As the target workpiece information was already contained in the $\Delta\boldsymbol{K}$-graph, it was not fed into the model as a 2nd input, different from Section 5.5.3.

7.5.2 Training method and hyperparameters

The DNNs were compiled in Python and trained using Keras with TensorFlow v2.2.0 as backend, and the computing facility had a NVIDIA Quadro RTX 6000 GPU with 24 GB of RAM memory. The training data for DNNs were all the tool paths learned from the RL algorithm, which were pre-processed to conform to the LSTM models and the labels (output features) were standardised to comparable scales. The tool path prediction with DNNs was configured to be a regression problem, for which the *Mean Square Error* (MSE) [56] was the objective function for DNN training. Adam algorithm [57], with default values of hyperparameters ($\beta_1, \beta_2, \epsilon$) in Keras, was used for optimisation. In addition, the learning rate η was set to exponentially decaying, from the initial learning rate η_0, along with training process, with the same decaying rate and decaying steps as in Section 4.4.2. The key training parameters are shown in Table 7-4, in which two amounts of training data are presented.

Table 7-4. Key parameters for DNN training.

Number of variants	Total data	Training data	Test data	Total time steps	Batch size	Initial learning rate η_0
25	1315	1184	131	63	32	$[10^{-6}, 10^{-2}]$
20	1012	910	102	55	32	$[10^{-6}, 10^{-2}]$

7.6 Learning results

7.6.1 Selection of reinforcement learning algorithm

Two categories of reinforcement learning algorithm, namely Q-learning and policy gradient algorithms, were compared in terms of their performances in tool path learning for the consecutive rubber-tool forming process. Subjected to the prohibitively expensive FE computation, a virtual environment was developed to imitate the rubber-tool forming behaviour as introduced in Section 7.4.2. A total of six RL algorithms were investigated with the data generated by the virtual environment, half of which belonged to Q-learning and the other half belong to policy gradient method. The most superior one determined in this study is to be implemented with FE environment and learn the optimal tool path using FE computational data. The same target workpiece, as shown in Figure 7-2, was used for tool path learning in this study.

The learning setups for all algorithms, including the transient (s_t, a_t, r_t, s_{t+1}), learning method and hyperparameters, are summarised in Section 7.4.1. An additional exploration rule, namely Heuristic 2, was implemented along with the ϵ-greedy policy for Q-learning algorithms. Figure 7-7 shows the performances of DQN, Double-DQN and Dueling-DQN trained under the exploration scheme with and without Heuristic 2, of which the termination step signifies the total punch steps spent to achieve the forming goal. The same learning rate (1×10^{-2}) and value function approximator (CNN) were applied to each training. It can be seen that the average termination step was reduced by approximately 40%, from 62 to 37, after introducing Heuristic 2 for exploration in the training of each algorithm. In addition, with Heuristic 2, the first applicable tool path was found more quickly in each case than those without Heuristic 2 by 9K, 2K and 3K training steps, respectively. Thus, because of the consistent improvement of learning efficiency from Heuristic 2 in each case, it was implemented for the training of all the Q-learning algorithms for the following results.

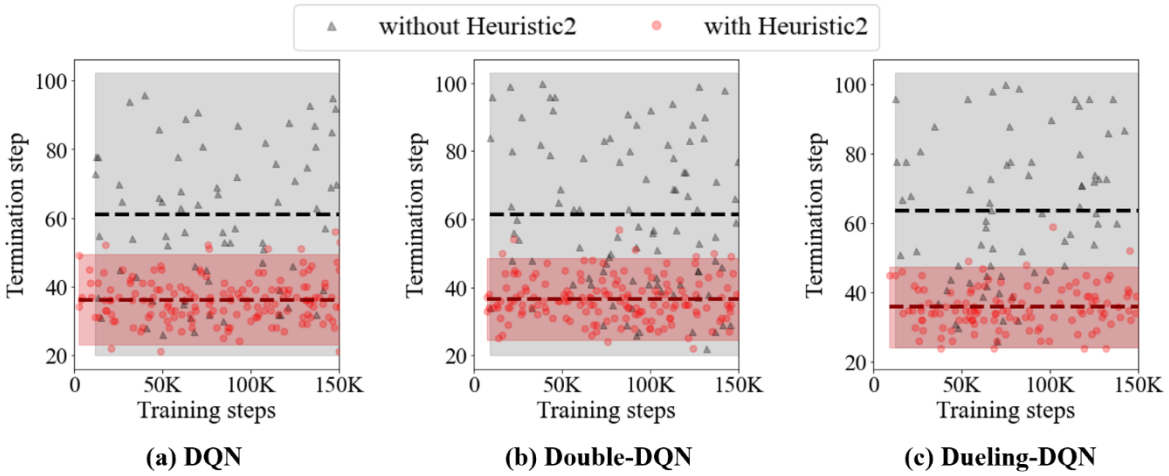


Figure 7-7. Comparison of the performances of three Q-learning algorithms trained at 10^{-2} learning rate, with and without heuristic, for tool path learning in terms of termination step. The upper and lower dashed lines denote the average termination step, estimated from Gaussian process regression, through the training process of the algorithms implemented with and without Heuristic 2, respectively. The shaded regions denote 95% confidence interval. The unit for training steps, K, denotes 10^3.

To comprehensively evaluate and compare the performance of the six RL algorithms in tool path learning, four performance factors with equal importance were raised, namely *first termination step* (1st Term. step), *converge speed* (Cvg. speed), *average converge termination step* (Avg. Cvg. Term. step) and *average termination frequency* (Avg. Term. Freq.). The first factor was quantified by the punch steps spent at the first time achieving the forming goal, which was used to evaluate the learning efficiency of each algorithm under the circumstance that no prior complete tool path planning experience was available and the agent learned the tool path from scratch. The 2nd and 3rd factors evaluated the learning progress and the learning results, and they were quantified by the first converged training step and the average termination step after convergence. The last factor described the learning steadiness in finding the tool path, which was computed as follow:

$$Avg.Term.Freq. = T_{total}/(S_{final} - S_{first}) \tag{7-1}$$

where T_{total} denotes the total times of termination during training, and S_{first} and S_{final} denote the training step where the first and final terminations occur, respectively.

The six algorithms were trained at four different learning rates using two action/policy function approximators, respectively, as described in Section 7.4.1 on the RL learning setup. The learning performance of each algorithm quantified by the four performance factors is summarised in Table 7-5 and Table 7-6. The learning results where no termination was found were omitted from the tables, except for Dueling-DQN which was designed to be a CNN with

shared convolutional layers and separate fully connected layers. For example, DDPG only managed to learn the tool path at the learning rate of 10^{-2} with MLP function approximator. The best tweaking results for the learning rate and function approximator for each algorithm are highlighted in bold.

Table 7-5. Performance comparison of the Q-learning algorithms for tool path learning. Term. denotes termination. Cvg. denotes converge. Freq. denotes frequency. Avg. denotes average. "-" denotes less than 3 terminations occur in the training.

Algorithm	Approximator	Learn rate	1st Term. step	Cvg. speed (unit: 10^3 steps)	Avg. Cvg. Term. step	Avg. Term. Freq. (unit: /10^3 steps)
DQN	MLP	10^{-1}	54	~ 23	53	0.14
		10^{-2}	**28**	**~ 2.5**	**32**	**1.24**
		10^{-3}	38	~ 25	35	2.18
		10^{-4}	52	~ 150	49	0.27
	CNN	10^{-1}	47	~ 20	35	1.35
		10^{-2}	**34**	**~ 4**	**35**	**1.09**
		10^{-3}	50	~ 22	34	2.08
		10^{-4}	52	~ 80	35	2.60
Double-DQN	MLP	10^{-1}	78	~ 43	78	-
		10^{-2}	**32**	**~ 3**	**34**	**1.28**
		10^{-3}	35	~ 80	38	2.23
		10^{-4}	71	~ 140	42	0.31
	CNN	**10^{-1}**	**37**	**~ 7**	**33**	**3.08**
		10^{-2}	33	~ 8	37	1.10
		10^{-3}	49	~ 80	36	2.77
		10^{-4}	50	~ 140	33	3.39
Dueling-DQN	**CNN**	10^{-1}	42	~ 25	42	-
		10^{-2}	45	~ 17	35	0.88
		10^{-3}	**41**	**~ 10**	**35**	**1.15**
		10^{-4}	51	~ 95	35	2.21

Table 7-6. Performance comparison of the policy gradient algorithms for tool path learning. Term. denotes termination. Cvg. denotes converge. Freq. denotes frequency. Avg. denotes average. "-" denotes less than 3 terminations occur in the training.

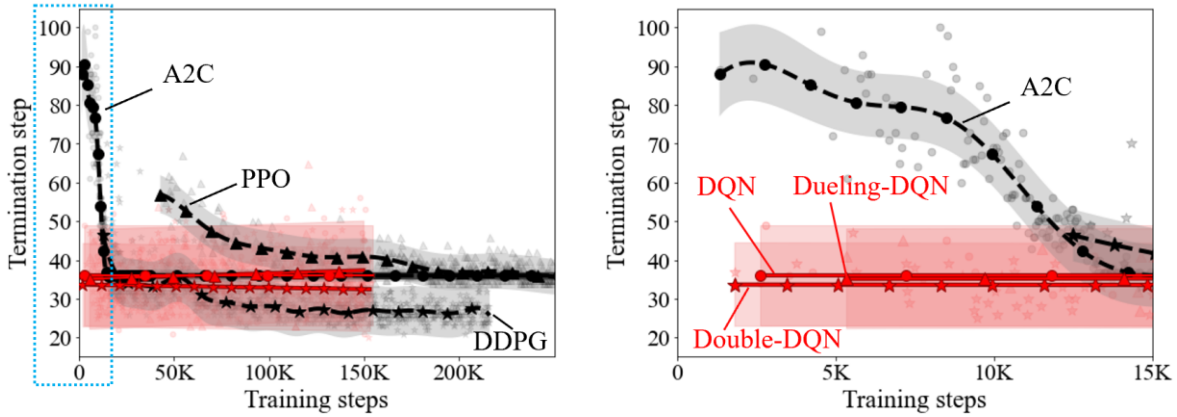
Algorithm	Approximator	Learn rate	1st Term. step	Cvg. speed (unit: 10^3 steps)	Avg. Cvg. Term. step	Avg. Term. Freq. (unit: /10^3 steps)
A2C	**MLP**	10^{-1}	78	~ 9	78	-
		10^{-2}	**89**	**~ 15**	**35**	**27.89**
		10^{-3}	92	~ 320	50	9.12
		10^{-4}	83	~ 36	90	0.82
	CNN	**10^{-2}**	**98**	**~ 100**	**62**	**16.25**
		10^{-3}	93	~ 95	69	14.08
		10^{-4}	100	~ 260	70	8.88
DDPG	**MLP**	**10^{-2}**	**49**	**~ 60**	**29**	**23.21**
PPO	MLP	10^{-1}	80	~ 133	80	-
		10^{-2}	**67**	**~ 700**	**41**	**21.98**
		10^{-3}	92	~ 450	54	0.005
		10^{-4}	64	~ 260	52	0.01
	CNN	10^{-1}	53	~ 55	62	0.05
		10^{-2}	**51**	**~ 200**	**35**	**28.25**
		10^{-3}	75	~ 3500	37	0.67
		10^{-4}	54	~ 2300	37	9.39

For Q-learning algorithms, the CNN function approximator was found to outperform the MLP one. Although they had close values of the first three performance factors, the average termination frequencies from the training of DQN and Double-DQN with CNN approximator were, in general, notably higher than those from algorithm trainings with MLP. It was seen that, with MLP, both DQN and Double-DQN cannot attain mere 0.3 terminations per thousand steps at 10^{-4} learning rate, and the latter terminated only once through the whole learning process at the learning rate of 10^{-1}. However, with CNN approximator, these two algorithms can both terminate steadily over one time per thousand steps at all learning rates. In regard to learning rate, 10^{-2}, 10^{-1} and 10^{-3} were respectively selected for the three Q-learning algorithms because of the evidently better results for the first three performance factors than the other choices of learning rate.

For policy gradient algorithms, MLP was selected as the function approximator for A2C and DDPG while CNN was selected for PPO, and the best learning results were found at learning rate of 10^{-2} for all three of them. It can be found that, compared to the Q-learning algorithms, the policy gradient algorithms tended to have a remarkably higher 1st termination step, of which those from A2C were approaching the maximum steps per episode (100). Although they converged to a comparable amount of average termination step to the Q-learning, they spent considerably more time in convergence, especially PPO which used over 200 thousand training steps (over 20 times longer than the Q-learning). In addition, the learning steadiness of the three policy gradient algorithms was poor, especially DDPG and PPO, although the best average termination frequency from them was over 9 times the best from the Q-learning. Trained with two different approximators and at four learning rates, DDPG only managed to learn the tool path once, which could be due to the reason that DDPG was developed for continuous action space problems.

Figure 7-8 shows the training process of the six RL algorithms, which were trained with the best hyperparameters from above. It can be seen that, unlike the Q-learning algorithms which almost instantly converged after a few terminations, the policy gradient ones had more discernible converging process. Although the tool paths learned from the policy gradient algorithms were about 1-3 times longer than those from the Q-learning at the start of training, they eventually converged to a comparable level of length. DDPG converged to the minimum average termination step of 29, however, its learning steadiness was the worst among all from Table 7-6. The Q-learning algorithms outperformed the policy gradient ones in general in terms

of the first termination step and convergence speed, and the Double-DQN preceded DQN and Dueling-DQN for its lower average converge termination step and marginally faster first termination. Thus, for the following results, Double-DQN was used to learn the optimal tool path.

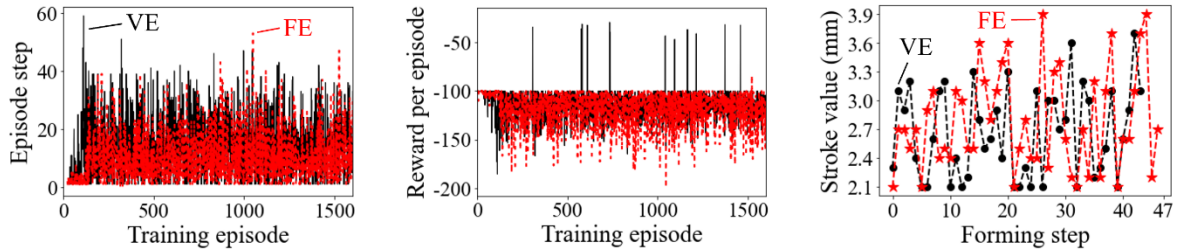


(a) Whole learning process **(b) Detailed view in dotted square**

Figure 7-8. Comparison of the performances of 6 reinforcement learning algorithms studied in this research in terms of termination step. The solid and dashed lines denote the average termination step, estimated from Gaussian process regression, through the training process of the algorithms. The shaded regions denote 95% confidence interval.

To assess the credibility of the algorithm selection study, the tool path learning processes and results of the Double-DQN, implemented with virtual environment (VE) and FE simulations, for the same target workpiece were compared in Figure 7-9. From Figure 7-9a and Figure 7-9b, the history of forming step and total rewards per episode from the learning using VE have the pattern which highly resembled those from the learning using FE simulations. The average episode step over the training process from VE was marginally higher than that from FE simulations by about 6 steps, while the average episode total rewards from the former was less than the latter by about 15. This phenomenon indicates that, with FE, the agent is more predisposed to overpunch (episode ends) with less efficient tool path at each episode than with VE. In addition, the first termination was about 1200 episodes slower and the total reward was about 30 less than those with VE. The practical rubber-tool forming behaviour was more complex and nonlinear than the VE imitated. From Figure 7-9c, both tool paths had similar forming pattern of alternatingly selecting small and large stroke values which had, on average, slightly increased throughout the tool path. Thus, in general, the virtual environment managed to imitate most of the forming behaviours in FE simulations, and the results from the pre-study

on algorithm selection performed with the virtual environment are convincing regarding their learning efficiency in tool path learning.

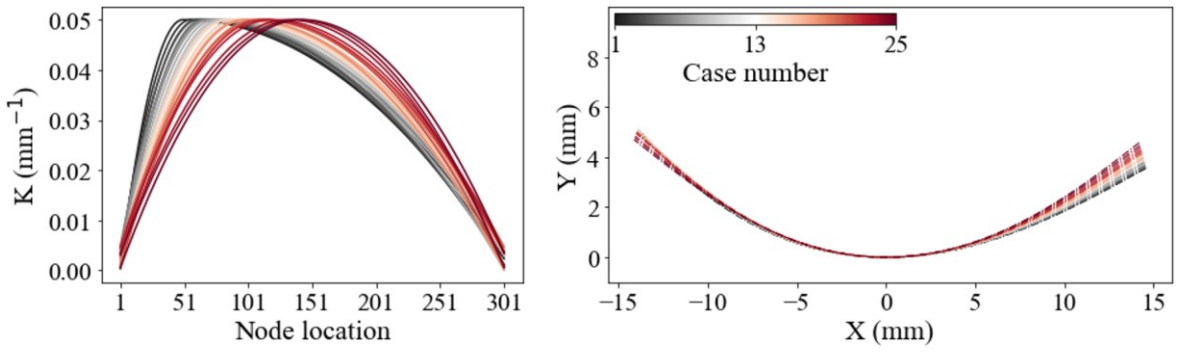


(a) History of episode step **(b) History of episode rewards** **(c) Tool path**

Figure 7-9. Comparison between RL using virtual environment (VE) and FE simulation environment (FE) in terms of (a) the history of steps in each episode, (b) the history of total rewards in each episode and (c) the tool path predictions.

7.6.2 Tool path learning for 25 workpieces using Double-DQN

From Section 7.6.1, Double-DQN was selected to learn the optimal tool paths for 25 variants of workpiece segments in Group 1, whose **K**-graphs and real-scale shapes are shown in Figure 7-10. The workpieces were deformed through the consecutive rubber-tool forming process, which was simulated through FE computations. The **K**-graphs were arbitrarily created with the method shown in Appendix B, and the real-scale shapes in Figure 7-10b were reconstructed from the **K**-graphs using constant initial interval between two contiguous node locations (0.1 mm).



(a) Target K-graphs **(b) Target workpiece shapes**

Figure 7-10. The (a) K-graphs and (b) workpiece shapes for all the target workpieces generated.

An exemplary Double-DQN learning process is shown in Figure 7-9a and Figure 7-9b, where the termination occurred at around episode 1500. It can be seen that the first 150 episodes ended with remarkably fewer forming steps than those thereafter, which is due to the effect of ϵ-greedy policy. Under this policy, the agent was more likely to randomly explore the search

space than following the online policy learned from existing forming experiences before the ϵ value decayed to 0.5, which led to quicker overpunch thus less steps per episode. To analyse the learning process in the light of effective forming progress, an exemplary learning process concerning the maximum $\Delta \boldsymbol{K}$ value at the end of each episode is shown in Figure 7-11. As the forming goal is to achieve a workpiece state where its $\max(|\Delta \boldsymbol{K}|) \leq 0.01$ mm^{-1}, the learning history of episode end maximum $\Delta \boldsymbol{K}$ can reflect the learning progress of effective tool path. From Figure 7-11, there is a clear trend that the maximum value of $\Delta \boldsymbol{K}$ at the end of each episode gradually decreased from 0.05 mm^{-1} at the start of learning to below 0.01 mm^{-1} at about episode 1350, where the termination occurred. Thus, the Double-DQN algorithm managed to learn the tool path by which the forming goal can be achieved.

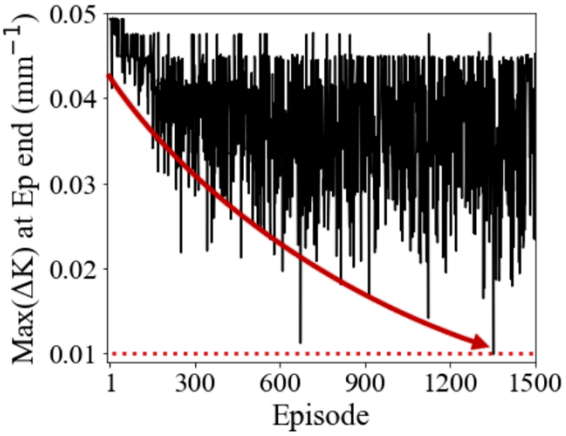
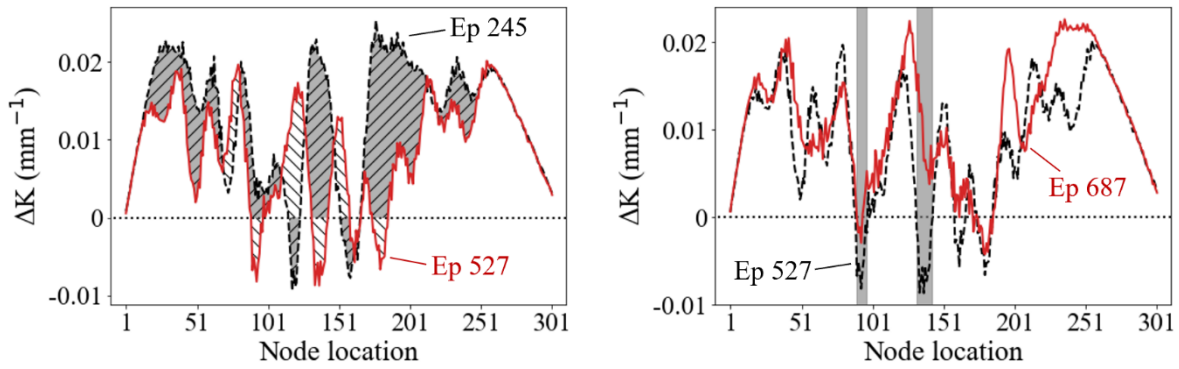


Figure 7-11. The history of the maximum ΔK value at the end of each episode (Ep) throughout the learning process. The arrow shows the learning progress of effective tool path.

In addition to the learning progress captured from maximum $\Delta \boldsymbol{K}$ curves, extra two self-learning characteristics of the tool path learning, which were measured from a more micro perspective than the former, were observed. Figure 7-12 shows two examples where the self-learning characteristics of tool path efficiency improvement and overpunch circumvention were captured, respectively. The three $\Delta \boldsymbol{K}$-graphs were collected from the workpiece deformed by the same number of punches at different episodes during a learning process. In Figure 7-12a, the shaded hatch denotes the total advantage from the workpiece state at episode 527 over that at episode 245 in terms of the shape difference from the target shape, which is measured by the area of hatch. In turn, the unshaded hatch represents the opposite. Thus, it is clear that the tool path planned at the recent episode was more efficient than the one at previous time by about 1.21 with reference to the net area of hatch (shaded area minus unshaded area), accounting for 11.8% of the initial $\Delta \boldsymbol{K}$-graph area. In Figure 7-12b, the shaded regions indicate two overpunch-

prone locations at episode 527, where the ΔK values were only within about 0.002 mm^{-1} away from the lower threshold (-0.01 mm^{-1}). Due to CPE2, the workpiece can be easily overpunched by deformation near the two locations. It was found that the agent selected smaller punch strokes at these two locations at episode 687, which circumvented the overpunch occurred in previous episodes.



(a) Improving tool path efficiency **(b) Circumvention of overpunch**

Figure 7-12. Examples showing self-learning characteristics of (a) improving tool path efficiency and (b) circumvention of overpunch. The results were from step 31 at three different episodes of the tool path learning process for a workpiece.

To evaluate the performance of the Double-DQN algorithm in extracting and learning abstract information during tool path learning, the representations in the last hidden layer of the Double-DQN model to the workpiece states, which the agent experienced throughout the learning process, were retrieved and reduced to two-dimensional embeddings through t-SNE technique [166]. The visualisation of these embeddings is shown in Figure 7-13, in which the embeddings are coloured according to selected stroke values by the agent. It can be seen that the CPE1, namely more prior deformation undergoes near the node location of interest results in larger punch stroke required to accomplish a certain change of shape at this location, was learned by the agent. From Figure 7-13, the $\Delta \boldsymbol{K}$-graphs ③, ④ and ⑤ were assigned relatively low value of stroke as there was no prior deformation to at least one side of the punch locations. In addition, larger stroke was assigned if the punch location was closer to the initial punch location 66, which is due to the higher amount of local shape difference. As the CPE1 escalated, higher stroke values were selected by the agent as shown by the rest of the $\Delta \boldsymbol{K}$-graphs except for ① and ②. It was also captured that, from ⑥ and ⑦, the CPE1 became more severe with larger nearby prior punches. Apart from CPE1, CPE2 was also captured by the $\Delta \boldsymbol{K}$-graphs ① and ②, where the effect was significantly more obvious than the one shown in Figure 7-4a. As indicated by the regions highlighted with circles in ① and ②, the ΔK values in these regions

were very close to the lower forming threshold, for which small strokes were assigned for them to prevent from overpunch. The punch effect at this region was reproduced as shown in Appendix C, where a mere increase of 0.1 mm of stroke can deteriorate the ΔK values to the left of the punch location by about 0.007 mm^{-1} and caused overpunch in this context. Overall, the similar workpiece states were clustered together and assigned by reasonable stroke values. The agent had a good understanding in tool path planning through learning the abstract representations.

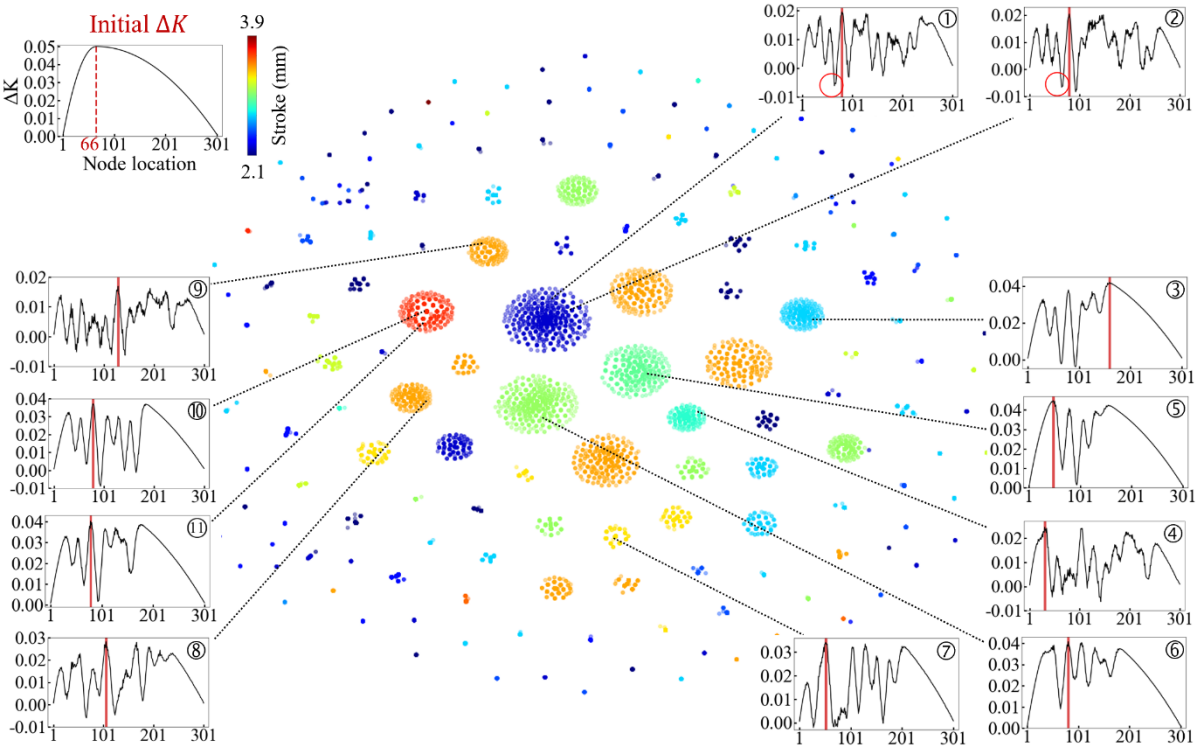
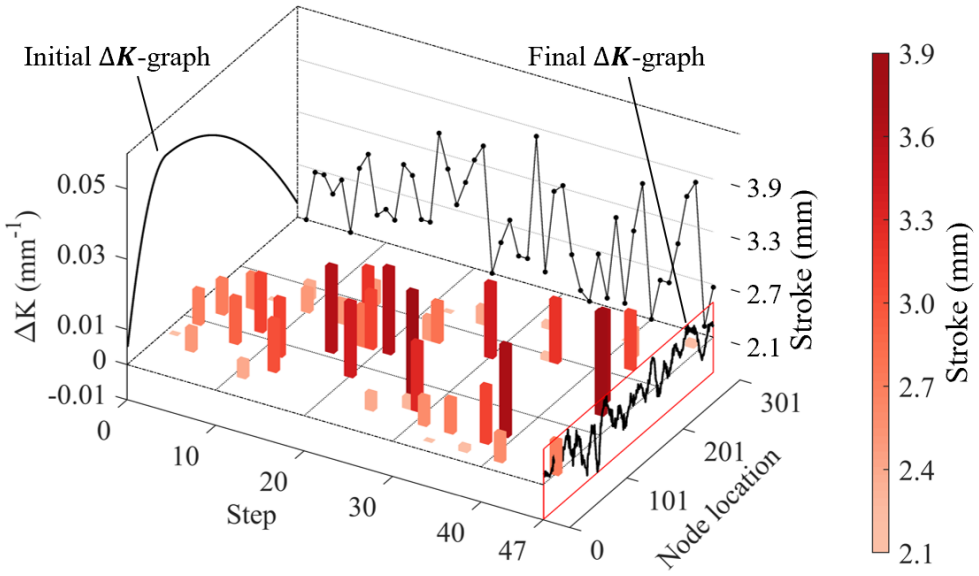


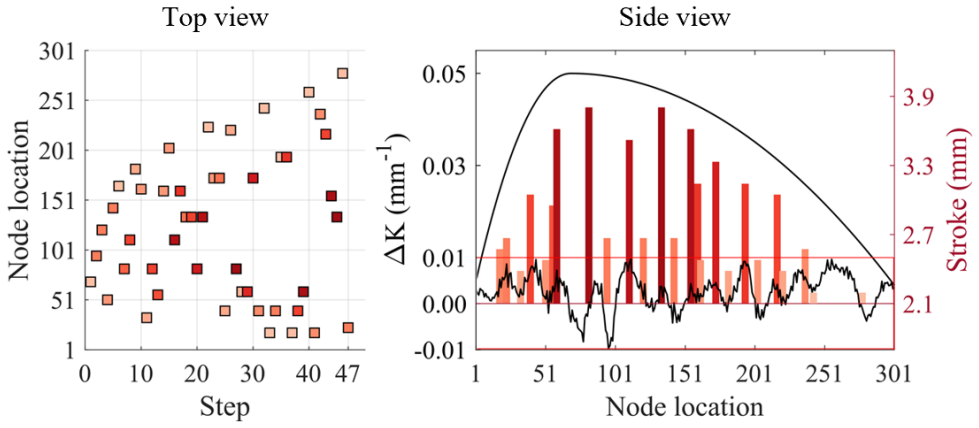
Figure 7-13. The two-dimensional embedding, generated through t-SNE, of the representations in the last hidden layer of the Double-DQN to workpiece states (ΔK-graphs) experienced during tool path learning. The points are coloured according to stroke values selected by the agent. The graph at the top left corner shows the initial ΔK-graph, and the axis labels of the other ΔK-graphs (numbered from ① to ⑪) are omitted for brevity. The vertical lines in the ΔK-graphs denote the punch locations.

Figure 7-14 shows an example of the tool path learned by the Double-DQN. In Figure 7-14a, the initial $\Delta \boldsymbol{K}$-graph between the blank sheet and the target workpiece was transformed to the final one (enclosed by a rectangle), where the ΔK values at all node locations were within the forming thresholds, by 47 forming steps. Due to the Heuristic 1 that the location with the highest ΔK value was selected as the punch location, the punch started from the location where the initial ΔK value was the highest (about 65) and diverged to both ends of the workpiece, as shown by the top view of Figure 7-14b. Lower values of stroke were assigned to diverging punches than those inside the divergence area due to the CPE1, which led to the repeatedly

alternating selection of small and large strokes along the forming progress in Figure 7-14a. As the forming progressed, the CPE1 escalated thus the larger stroke values were selected at later steps (after step 15) of the tool path than those at start. It is also worth noting that, from the side view in Figure 7-14b, large strokes were concentratedly assigned to punch locations with high initial ΔK values and descended to those with low ones.



(a) Tool path: punch location and stroke at each step



(b) Different views

Figure 7-14. An example of (a) tool path learned by RL and (b) its top view and side view. The height of the bars in (a) were proportionally decreased for better visualisation.

Figure 7-15 presents the deformation process of a workpiece from blank sheet to its target shape following the tool path learned from the Double-DQN and the dimension error (the geometry difference in Y-direction) between the target workpiece and the one after all punches. The target shape in the real-scale graph was reconstructed from the target **K**-graph, of which

the same interval of 0.1 mm between two contiguous node locations along the deformed workpiece was used for reconstruction. It can be seen that the final shape of this deformed workpiece was in a good agreement with its target shape, with a maximum dimension error of just above 0.2 mm. The dimension error was at its minimum in the middle of the workpiece, from which the error increased to both ends due to the accumulation of shape difference. The average maximum dimension error for the 25 variants using the tool paths learned by the Double-DQN algorithm was 0.26 mm.

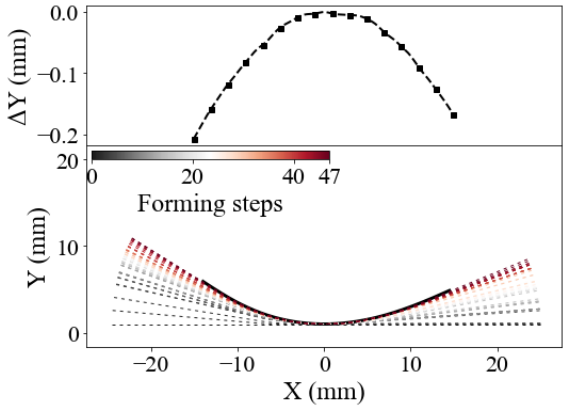


Figure 7-15. An example of workpiece deformed by the tool path predicted by the Double-DQN. Top: the dimension error between the target workpiece and the final workpiece deformed along the predicted tool path. Bottom: the workpiece shape at each forming step (dotted line) compared to its target shape (solid line). Forming step of 0 denotes blank sheet.

7.6.3 Tool path generalisation using supervised learning model

Through Double-DQN algorithm, the tool paths for the 25 variants of workpiece (shown in Figure 7-10) segment were learned, whose length (total punch steps) for each variant is shown in Figure 7-16. The tool path lengths varied from 44 to 63, with most lying around 52.

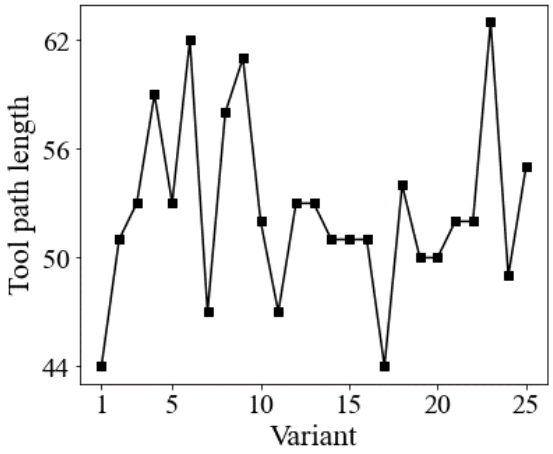


Figure 7-16. The length of the tool path learned through Double-DQN for each variant of workpiece segment.

To learn the intrinsic efficient forming pattern for these workpiece variants (Group 1), the supervised learning model was used for training with the tool path data for the 25 variants. As introduced in Section 7.5.1, three LSTMs, which respectively used VGG16, ResNet34 and ResNet50 as the feature extractor, were investigated. The training data were the 25 tool paths pre-processed to the data format consistent with the input and output of the CNN LSTMs, with a total of 1315 data. These data were uniformly split into 90% for training and 10% for testing, and the other key training parameters are presented in Table 7-4. The training processes of the three models are shown in Figure 7-17 by generalisation loss (test loss) history, which would end early if the generalisation loss tended to increase [56]. In addition to training models using the total amount of 25 tool paths, the VGG16 LSTM was also trained with only 20 tool paths to study the effect of training data on the learning performance. The 20 tool paths were evenly sampled from the original 25 paths to avoid massive data missing, and the maximum tool path length among the 20 paths was 55. The generalisation loss has been de-standardised to stroke unit (mm), and the losses from the three models all converged to a comparable level of 0.25 mm, except for the VGG16 LSTM trained with 20 tool paths whose loss converged to about 0.33 mm. Thus, more training data was seen to improve the generalisation, which could be due to that more exhaustive data help to generalise the forming pattern during training.

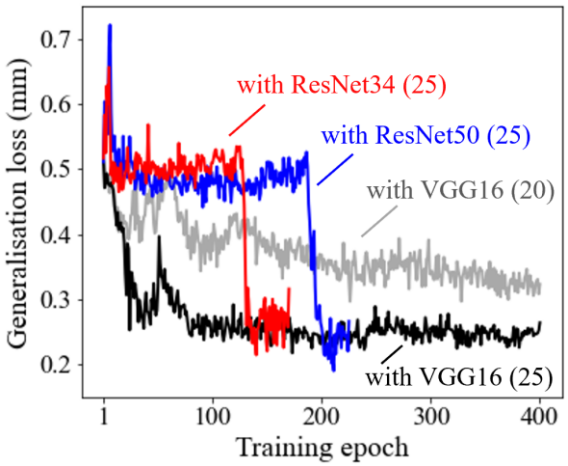


Figure 7-17. The generalisation loss curves along the training processes of the LSTM models with VGG16, ResNet34 and ResNet50 as the feature extractor, respectively. The number 25 and 20 in the parenthesis denote the amount of tool paths used for training.

Figure 7-18 shows the prediction results for the same test workpiece from the three models trained with 25 tool paths and the VGG16 LSTM trained with 20 tool paths.

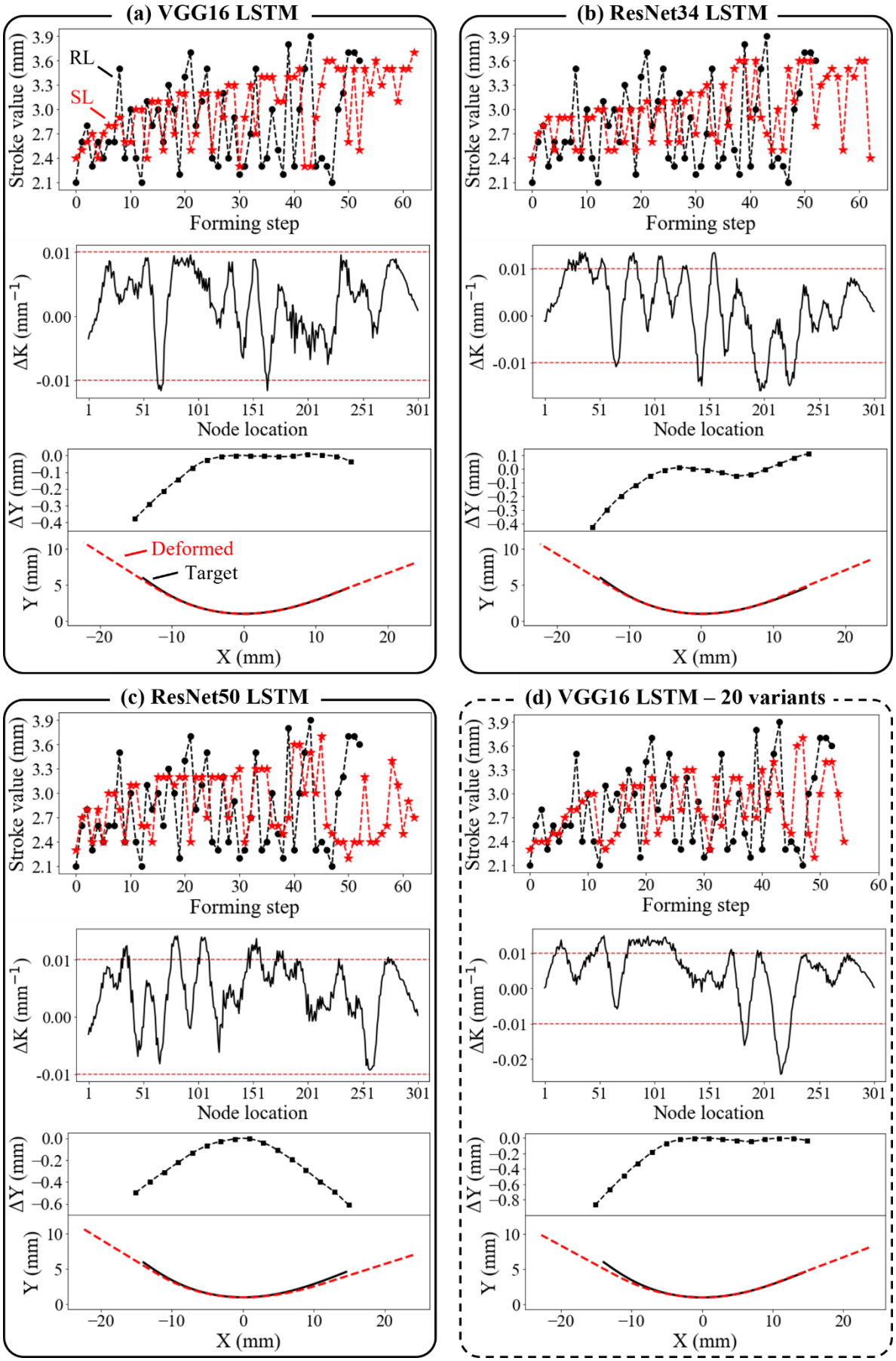

(a) VGG16 LSTM
(b) ResNet34 LSTM
(c) ResNet50 LSTM
(d) VGG16 LSTM – 20 variants
Stroke value (mm)
Forming step
RL
SL
ΔK (mm⁻¹)
Node location
ΔY (mm)
Y (mm)
Deformed
Target
X (mm)

Figure 7-18. The learning performance of LSTMs trained with 25 tool paths data (solid squares) and 20 tool paths data (dashed squares) on a test workpiece. For the former, the prediction results from LSTMs with (a) VGG16, (b) ResNet34 and (c) ResNet50 are presented, while for the latter, that from (d) VGG16 LSTM are presented. The prediction results include three parts as followed. Top: the tool path prediction from LSTMs (SL) and its comparison to the tool path from reinforcement learning (RL); Mid: the final ΔK-graph after deformation; Bottom: the dimension error and the comparison between the deformed workpiece shape and its target.

As the total time steps of the LSTMs were set to be the same as the maximum forming steps in the training data, the tool path prediction terminated at 63 and 55 forming step for models trained with 25 and 20 tool paths, respectively. It can be seen that the tool path predictions for the workpiece from the three supervised learning models trained with different amount of data all agreed well with the tool path learned through reinforcement learning. The punch started from small values of stroke and alternatingly selected small and large strokes along the forming progress, and the values of large strokes gradually increased as the CPE1 escalated in the forming process. However, it is worth noting that the ResNet50 LSTM tended to predict successive lower values of strokes, near the end of forming (from step 46 to 57), than those predicted by the other models.

With regard to the final $\Delta \boldsymbol{K}$-graph of the test workpiece deformed through the tool path predicted by the LSTMs, the VGG16 LSTM trained with 25 tool paths was the most superior among all models, whose level of forming goal achievement ($G = 1 - \Delta \boldsymbol{K}_{final}^{out\ THLD} / \Delta \boldsymbol{K}_{initial}^{out\ THLD}$, THLD denotes threshold) was up to 99.9%. However, the level of goal achievement of the other two models trained with 25 tool paths just reached 97%, and the LSTM trained with 20 tool paths only achieved 95%. From the final $\Delta \boldsymbol{K}$-graphs from models trained with 25 tool paths in Figure 7-18, the one from the VGG16 LSTM was seen to have only two negligible overpunch at location 66 and 161, of which location 66 was the first punch location in the tool path and the overpunch was due to the accumulation of CPE2 near this location through the rest of the tool path. However, multiple evident overpunches and short (insufficient) punches were found in the $\Delta \boldsymbol{K}$-graph from the ResNet34 LSTM and short punches in that from the ResNet50 LSTM. This is due to the over- and under-estimation of stroke values in the tool path, from the models, at the locations where overpunches and short punches occurred, and the short punches from ResNet50 LSTM could be caused by the massive punch steps of low stroke values near the end of forming. As for the $\Delta \boldsymbol{K}$-graph from the VGG16 LSTM trained with 20 tool paths, the result was even worse than those trained with 25 tool paths. It was seen to have the worst overpunch at node location 213 among the four cases, and there was a continuous short punch region from

location 72 to 122, indicating a consistent underestimation of stroke values for punches in this region. The consistent underestimation could be caused by the less training data, which led to the lack of useful tool path data for this region.

In terms of the final geometry difference between the deformed workpiece and its target shape, the tool path predicted by the VGG16 LSTM preceded those from the other models, through which the maximum dimension error was about 0.37 mm. However, the dimension errors resulted from other models were much higher, especially for ResNet50 LSTM and the one trained with less data whose predictions led to over 0.6 mm and 0.8 mm of dimension error, respectively. In addition, the final workpiece shapes from these two models had much more visible deviation from their targets than those from VGG16- and ResNet34- LSTMs trained with 25 tool paths.

It is worth noting that, although the $\Delta\boldsymbol{K}$-graph from the ResNet34 LSTM was not as good as the one from the VGG16 LSTM according to the level of forming goal achievement G, the tool paths from both models resulted in comparable results of the final dimension error. On the other hand, good goal achievement does not entail good dimensional accuracy. For example, the right half of workpiece shape from ResNet50 LSTM remarkably differed from its target although its corresponding $\Delta\boldsymbol{K}$-graph had a good goal achievement, which is due to the excessively more area of $\Delta\boldsymbol{K}$-graph above the X-axis than that below it. The four models, the VGG16-, ResNet34- and ResNet50 LSTM trained with 25 tool paths and the VGG16 LSTM trained with 20 tool paths, were re-evaluated on 10 arbitrary variants, and the average level of forming goal achievement $\bar{G}$ from them was 99.54%, 96.86%, 97.15% and 97.19%, respectively. With regard to the maximum dimension error, the average value from the four models was 0.45, 0.40, 0.63 and 0.58 mm, respectively. It was seen that, although VGG16 LSTM had remarkably better goal achievement than the ResNet34 LSTM, it yielded slightly larger dimension error. This indicates that the overpunch and short punch in terms of the $\Delta\boldsymbol{K}$ thresholds can, to some extent, contribute to the final forming results. The results entail that moderate compromise of workpiece curvature smoothness could bring more effective tool path planning behaviour in terms of dimensional accuracy. Multi-objective optimisation could be considered in the future for learning the optimal trade-off between the final curvature smoothness and the dimensional accuracy. Thus, attaining the best level of goal achievement

and leading to a high dimensional accuracy, the VGG16 LSTM trained with larger amount of data had the most superior performance in tool path learning generalisation.

To compare the tool path planning performance of the proposed generalisable strategy and the method exploiting pure reinforcement learning technology, a well-trained Double-DQN model was used for stroke prediction of the first punch for variants of workpiece with different initial $\Delta \boldsymbol{K}$-graphs (i.e., different target shapes), including the target shape it was trained for. Figure 7-19 shows the $\Delta \boldsymbol{K}$-graphs after the first punch of these workpieces with the stroke predicted by the Double-DQN, of which the node locations where the troughs reside indicate the punch locations. It can be seen that most of the stroke predictions for new variants were uncharacteristically large, which caused significant overpunch at the very first forming step. This indicates that the Double-DQN trained for the tool path learning of a certain target shape cannot be used to predict the tool path for different target workpieces, and the reinforcement learning process has to be gone through again for new applications.

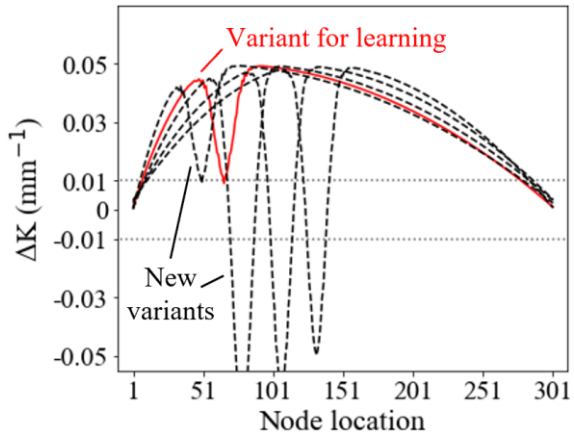


Figure 7-19. Evaluation of the pure RL strategy by assessing the tool path prediction results from the trained Double-DQN model for new variants of workpiece. The ΔK-graphs were acquired after the first punch, predicted by the RL model, for the variant used for tool path learning through RL (solid line) and new variants that were never seen in the learning process (dashed line).

7.7 Case study verifying the generalisable tool path planning strategy

To evaluate the generalisable tool path planning strategy presented in Figure 6-9, a new target workpiece of length 90.2 mm was arbitrarily generated as shown in Figure 7-20. The target workpiece was first digitised to its initial $\Delta \boldsymbol{K}$-graph using 0.1 mm interval between two contiguous node locations, with 903 node locations in total. The $\Delta \boldsymbol{K}$-graph can be segmented to 3 Group 1 segments, A, B and C, which were never seen in the training process of the

proposed strategy. With the trained supervised learning model (VGG16 LSTM), the forming tool path for each of the segment was predicted and aggregated to the entire tool path for the target workpiece. It can be seen that the deformation took place segment-wise, and the final $\Delta \boldsymbol{K}$-graph resided well in the threshold region with the level of forming goal achievement of 99.87%.

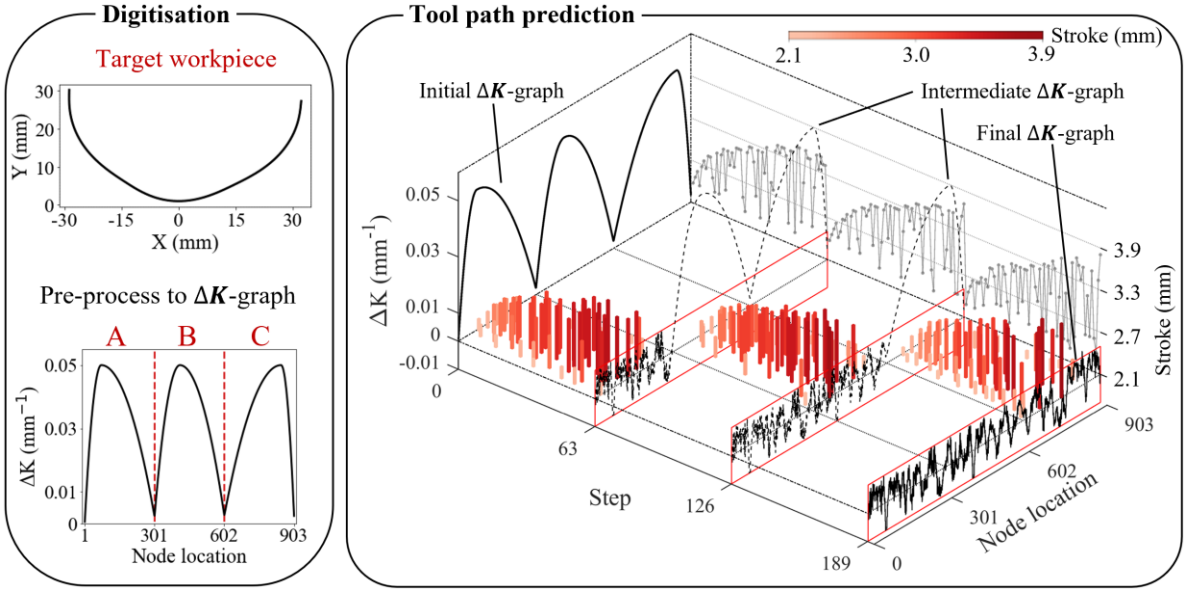


Figure 7-20. The entire tool path for a new workpiece, which is aggregated by the subpaths predicted by the VGG16 LSTM for each segment of the workpiece. The three segments were never seen in the LSTM training.

Figure 7-21 presents the workpiece shape after deformation, computed by FE, through the generalisable tool path planning strategy and its target shape. From Figure 7-21a, due to the accumulation of $\Delta \boldsymbol{K}$-graph area above the X-axis near the junction location (location 301 and 602) between two segments, there was a visible deviation between the deformed workpiece shape and its target, with a maximum dimension error of about 1.8 mm. With two supplementary punches at the two junction locations, the deformed workpiece shape had a clear approaching to the target shape, with a maximum dimension error of about 1 mm. Since the punch location was at the end of each segment where CPE1 was not escalated, small stroke values were selected for the two supplementary punches, which did not cause overpunch. Thus, the generalisable tool path planning strategy successfully yielded the final workpiece shape within a dimension error of 2%. Due to the error accumulation brought in by the junction area, the deformed workpiece shape can be further improved by a few supplementary punches in this area.

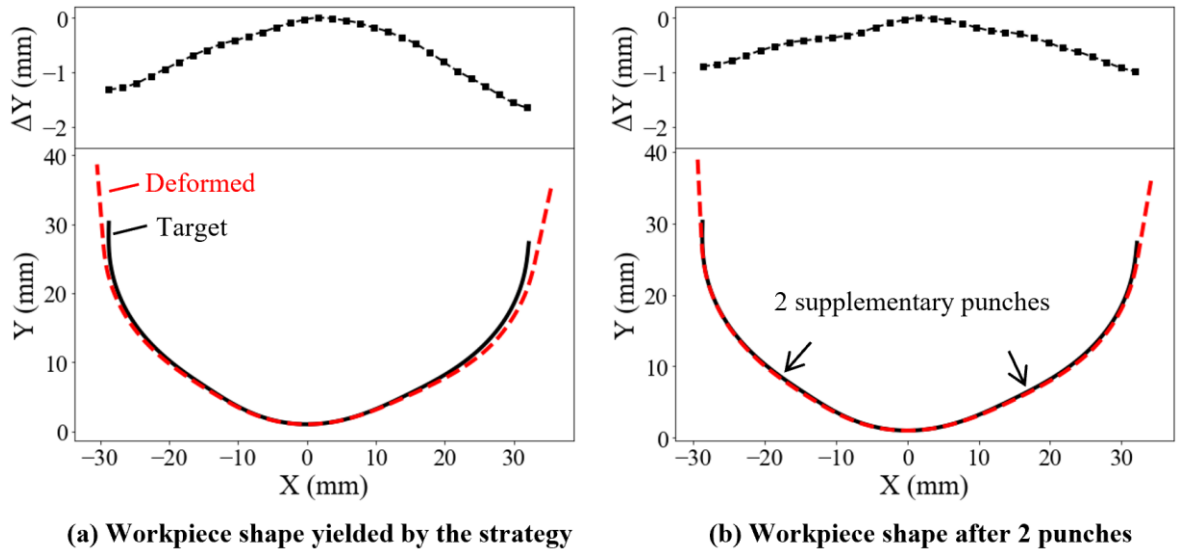


Figure 7-21. The comparison between the workpiece shape deformed through the generalisable tool path planning strategy and its target shape. (a) deformed workpiece shape yielded by the strategy and (b) workpiece shape after two supplementary punches near the junctions of the three workpiece segments.

7.8 Summary

In this chapter, a generalisable tool path planning strategy, through deep reinforcement and supervised learning technologies, has been developed for tool path planning of arbitrary sheet metal parts formed by a consecutive free-form sheet metal stamping process. The strategy factorises the entire target workpiece into groups of segments according to predetermined shape features and generates the optimal tool path through dynamic programming.

Q-learning algorithms are superior to policy gradient algorithms in tool path planning of consecutive free-form sheet metal stamping process, of which Double-DQN precedes DQN and Dueling-DQN. In addition, the forming heuristic is corroborated to further improve the Q-learning performance.

Conferred by deep reinforcement learning, the generalisable tool path planning strategy manifests self-learning characteristics. Over the learning process, the tool path plan becomes more efficient and learns to circumvent overpunch-prone behaviours. With Double-DQN, the tool path for a consecutive free-form sheet metal stamping process can be successfully acquired, with the dimension error of the deformed workpiece below 0.26 mm (0.87%).

The efficient forming pattern for a group of workpiece segments have been successfully generalised using deep supervised learning models. The VGG16 LSTM precedes ResNet34-

and ResNet50 LSTMs in the tool path learning generalisation, although they have comparable average generalisation loss. The VGG16 LSTM manages to predict the tool path for 10 test variants, with an average level of forming goal achievement of 99.54% and a dimension error of the deformed workpiece below 0.45 mm (1.5%). However, the pure reinforcement learning method cannot generalise plausible tool paths for completely new workpieces.

The generalisable tool path planning strategy successfully predicts the tool path for a completely new workpiece, which has never been seen in its previous learning experience. The level of goal achievement reaches 99.87% and the dimension error of the deformed workpiece is 2%. The dimension error could be reduced to about 1.1% with two small supplementary punches near the junctions of the workpiece segments.

Chapter 8

Conclusions and future work

8.1 Conclusions

Production flexibility in the development of contemporary sheet metal forming technology is becoming increasingly important, and the extraordinarily high capital cost on customised punch and dies, especially for extra-large components manufacturing, also leads to expensive prototyping and arduous exploitation of novel sheet metal designs. Thus, an alternative approach, flexible forming, is developed in this thesis to replicate the workflow of a blacksmith, which can form parts of different geometries using several general-purpose tools. By harnessing state-of-the-art deep learning technologies, an intelligent flexible sheet metal forming system has been established, for the first time, for tool path planning of arbitrary sheet metal parts in this thesis. The system is composed of main modules for tool path planning and subsidiary modules for auxiliary tasks, of which the modules for tool path planning are developed through DSL and DRL technologies. A generalisable tool path planning strategy has been proposed and verified to significantly extend the applicability of the intelligent forming system. The following conclusions can be drawn:

- The concept of intellectualising sheet metal forming processes with machine learning technology has been successfully embedded in an intelligent flexible sheet metal forming system, which exploits DSL and DRL techniques in different modules to generate and optimise tool paths for sheet metal parts. The tool path planning accuracy and efficiency is ensured through a dual modular redundancy-akin approach.
- Supervised learning models, including a convolutional neural network and a SVR model, have been used to model the intrinsic relationship between the punch stroke and its resulting morphed sheet metal geometry after springback in bending processes. The network model is seen to have consistently high generalisation accuracy with

decreasing amount of training data, while the performance of SVR deteriorates. By modelling the bridge between target workpiece shape and its corresponding forming parameter, issues concerning springback in the process design can be circumvented. Thus, DNN approach provides an alternative method for springback compensation with shorter development cycle and less computational requirement than traditional compensation methods.

- A novel theory-guided regularisation method for DNN training has been proposed, utilising Swift's law as guidance. The method regularises the learning of two extra learning tasks (local stress and strain prediction) to improve the learning of the primary task (forming parameter prediction). TG-DNN outperforms data-driven DNN in process parameter prediction in severely scarce data condition, which has been verified through experiments with different data structures, sheet metal materials and applications. In addition, the TG-DNN manifests consistently high generalisation accuracies in these experimental conditions, especially in the one with asymmetric training data structure where a large domain of training data is missing. The generalisation losses of extra tasks prediction from TG-DNN are evidently less than those from data-driven DNN in all experiments, which demonstrates the utility of theory-guided regularisation method.
- A consecutive rubber-tool forming process has been proposed with its FE model. A recursive tool path prediction framework has been proposed for tool path planning of this process, of which a deep learning model is used for tool path learning. The recursive framework adopts the traditional sheet metal shaping convention, which selects forming operations based on the shape difference between the current and target workpiece shape. This framework has been successfully implemented for tool path prediction with DNNs and attains over 90% prediction accuracy.
- Three series of models have been investigated for tool path learning within the recursive prediction framework, namely single feature extractor, cascaded network and LSTM, which cover various architectures of CNN, RNN and autoencoder. The workpiece geometry is pre-processed to image-based representation to fully exploit the feature learning capability of CNN. cGAN is found to have the most superior image reconstruction performance among all models investigated, and CNN LSTM is verified

to be the most superior in tool path planning with over 90% generalisation accuracy achieved.

- An end-to-end DRL approach, with DQN algorithm, has been investigated to learn the optimal movement of a single punching operation, including a sequence of punch translation and punching speed selection. This approach iteratively predicts the forming action until a prescribed workpiece geometry, which is deformed by a single punch at a random location, has obtained. After training, the DRL approach successfully learns an optimal tool path towards the punch location and selects the correct punching speed, which reveals the feasibility of intellectualising traditional sheet metal flexible forming processes without any human intervention.
- A generalisable strategy has been proposed for tool path planning of sheet metal parts of arbitrary shape with no metal forming knowledge in prior. This strategy factorises the workpiece into segments and classify them into three groups. The tool path for each segment is learned using DRL, and the efficient forming pattern for each group of segments is generalised using DSL. The strategy is verified capable of self-improving tool path efficiency and circumventing infeasible forming paths over time. The strategy successfully predicts the tool path for a completely new workpiece with 2% dimension error, and it is verified to have significantly better generalisation capability than pure DRL approach through a case study.
- For the DRL algorithms in the path planning strategy, six algorithms have been implemented and compared in terms of their tool path learning performance, including DQN, Double-DQN and Dueling-DQN from Q-learning algorithms and A2C, DDPG and PPO from policy gradient algorithms. Q-learning algorithms are found to precede policy gradient ones, and Double-DQN is demonstrated to be the most superior, with its learned tool path yielding below 1% dimension error.
- For the DSL models in the path planning strategy, three CNN LSTMs have been implemented and compared in terms of their path learning generalisation performance, including VGG16-, ResNet34- and ResNet50 LSTM. The VGG16 LSTM is verified to outperform the other two, with its predicted tool paths for test workpieces yielding only 1.5% average dimension error.

8.2 Future work

This thesis has established a simulation-based intelligent flexible sheet metal forming system, with dedicated DNNs and tool path planning strategy developed. Some future work can be performed to further extend its applicability and promote its industrial applications:

- The Process FE Module in the intelligent system can be replaced by a well-developed surrogate model, as illustrated in Figure 3-5, to facilitate the tool path optimisation process in the Process Learning Module. Through uniform data sampling within the allowed stroke range and the workpiece punch area curvature data domain, a neural network based surrogate model can be developed with favourable prediction accuracy. Using this surrogate model, optimal tool paths can be learned for new sheet metal parts in short time with forming actions restricted in the sampled data domain. In addition, once the surrogate model has been calibrated with experimental data, the intelligent system can be further applied to practical sheet metal tool path planning problems.
- The theory-guided learning method proposed in this thesis is instantiated for single deformation step prediction using a CNN. It can be further promoted to improve the forming sequence learning with LSTM and alleviate the training data demand for tool path planning. Harder regularisation methods can be attempted, by either introducing new/more relevant theories or approximation equations describing the deformation at the punching area.
- The generalisable tool path planning strategy can be extended to multi-objective optimisation to consider a trade-off between the final curvature smoothness and the dimensional accuracy. With moderate compromise of curvature smoothness, more efficient tool path in terms of dimensional accuracy could be yielded. Furthermore, the tool path learning strategy can also include a two-pass process, with the first pass shortly obtaining a coarse solution and the second yielding the fine solution. By expanding the optimisation scope from local segment to the whole workpiece, the tool path optimisation results are more likely to be the global optimum. By restricting the input curvature shape of the second pass, the generalisability of this strategy can be preserved.

References

[1] Audi puts steel back in the new A8. https://www.greencarcongress.com/2017/05/20170512-audia8.html (accessed April 11, 2022).

[2] R.J.H. Wanhill, Aerospace applications of aluminum–lithium alloys, in: Aluminum-Lithium Alloy., Elsevier, 2014: pp. 503–535. https://doi.org/10.1016/B978-0-12-401698-9.00015-X.

[3] High-speed rail transport train railway bullet train. https://www.kissclipart.com/high-speed-rail-transport-train-railway-bullet-train-4l22s1/ (accessed April 11, 2022).

[4] K. Zheng, D.J. Politis, L. Wang, J. Lin, A review on forming techniques for manufacturing lightweight complex—shaped aluminium panel components, Int. J. Light. Mater. Manuf. 1 (2018) 55–80. https://doi.org/10.1016/j.ijlmm.2018.03.006.

[5] T. Altan, A.E. Tekkaya, Sheet metal forming: Processes and appliations, Materials Park: ASM International, 2012.

[6] Z. Gu, S. Li, L. Zhao, L. Zhu, G. Yu, Finite element simulation and experimental investigations of cold stamping forming defect of A588-A thick weathering steel bogie lower cover, Int. J. Adv. Manuf. Technol. 104 (2019) 1275–1283. https://doi.org/10.1007/s00170-019-04148-5.

[7] J. Zhou, X. Yang, B. Wang, W. Xiao, Springback prediction of 7075 aluminum alloy V-shaped parts in cold and hot stamping, Int. J. Adv. Manuf. Technol. 119 (2022) 203–216. https://doi.org/10.1007/s00170-021-08204-x.

[8] K. TAKATA, Warm forming of aluminum alloys, Nippon Steel Tech. Rep. (2013) 104–109.

[9] A. Wang, K. Zhong, O. El Fakir, J. Liu, C. Sun, L.-L. Wang, J. Lin, T.A. Dean, Springback analysis of AA5754 after hot stamping: experiments and FE modelling, Int. J. Adv. Manuf. Technol. 89 (2017) 1339–1352. https://doi.org/10.1007/s00170-016-9166-3.

[10] X. Fan, Z. He, K. Zheng, S. Yuan, Strengthening behavior of Al–Cu–Mg alloy sheet in hot forming–quenching integrated process with cold–hot dies, Mater. Des. 83 (2015) 557–565. https://doi.org/10.1016/j.matdes.2015.06.058.

[11] Y. Hu, Interactive modelling of the evolutionary tribological phenomena in cold stamping, Imperial College London, 2018. https://doi.org/https://doi.org/10.25560/69540.

[12] H. Karbasian, A.E. Tekkaya, A review on hot stamping, J. Mater. Process. Technol. 210 (2010) 2103–2118. https://doi.org/10.1016/j.jmatprotec.2010.07.019.

[13] M. Merklein, J. Lechler, Investigation of the thermo-mechanical properties of hot stamping steels, J. Mater. Process. Technol. 177 (2006) 452–455. https://doi.org/10.1016/j.jmatprotec.2006.03.233.

[14] K. Mori, Review of shearing processes of high strength steel sheets, J. Manuf. Mater. Process. 4 (2020) 54. https://doi.org/10.3390/jmmp4020054.

[15] T. Trzepieciński, Recent developments and trends in sheet metal forming, Metals (Basel). 10 (2020) 779. https://doi.org/10.3390/met10060779.

[16] D.R. Cooper, K.E. Rossie, T.G. Gutowski, The energy requirements and environmental impacts of sheet metal forming: An analysis of five forming processes, J. Mater. Process. Technol. 244 (2017) 116–135. https://doi.org/10.1016/j.jmatprotec.2017.01.010.

[17] G.S. Daehn, A. Taub, Metamorphic manufacturing: The third wave in digital manufacturing, Manuf. Lett. 15 (2018) 86–88. https://doi.org/10.1016/j.mfglet.2018.02.014.

[18] D.T. Bowen, I.M. Russo, C.J. Cleaver, J.M. Allwood, E.G. Loukaides, From art to part: Learning from the traditional smith in developing flexible sheet metal forming processes, J. Mater. Process. Technol. 299 (2022) 117337. https://doi.org/10.1016/j.jmatprotec.2021.117337.

[19] A. Qattawi, A. Mayyas, S. Dongri, M. Omar, Knowledge-based systems in sheet metal stamping: a survey, Int. J. Comput. Integr. Manuf. 27 (2014) 707–718. https://doi.org/10.1080/0951192X.2013.834463.

[20] B.T. Cheok, J.Y. Li, A.Y.C. Nee, Integrated feature-based modelling and process planning of bending operations in progressive die design, Int. J. Adv. Manuf. Technol. 20 (2002) 883–895. https://doi.org/10.1007/s001700200212.

[21] X. Shi, J. Chen, Y. Peng, X. Ruan, Development of a knowledge-based process planning system for an auto panel, Int. J. Adv. Manuf. Technol. 19 (2002) 898–904. https://doi.org/10.1007/s001700200102.

[22] J. Zheng, Y. Wang, Z. Li, KBE-based stamping process paths generated for automobile panels, Int. J. Adv. Manuf. Technol. 31 (2006) 663–672. https://doi.org/10.1007/s00170-005-0239-y.

[23] Sheet metal shaping. https://www.facebook.com/The-Garage-Metalshaping-Studio-222352857822743/photos (accessed July 20, 2022).

[24] B. Ilangovan, R.P. Monfared, M. Jackson, An automated solution for fixtureless sheet metal forming, Int. J. Adv. Manuf. Technol. 82 (2016) 315–326. https://doi.org/10.1007/s00170-015-7366-x.

[25] R. Singh, Heat treatment of steels, in: Appl. Weld. Eng., Elsevier, 2016: pp. 111–124. https://doi.org/10.1016/B978-0-12-804176-5.00012-8.

[26] K.O. Pedersen, H.J. Roven, O.-G. Lademo, O.S. Hopperstad, Strength and ductility of aluminium alloy AA7030, Mater. Sci. Eng. A. 473 (2008) 81–89. https://doi.org/10.1016/j.msea.2007.03.089.

[27] H. Takuda, K. Mori, I. Masuda, Y. Abe, M. Matsuo, Finite element simulation of warm deep drawing of aluminium alloy sheet when accounting for heat conduction, J. Mater. Process. Technol. 120 (2002) 412–418. https://doi.org/10.1016/S0924-0136(01)01180-3.

[28] G. Palumbo, L. Tricarico, Numerical and experimental investigations on the Warm Deep Drawing process of circular aluminum alloy specimens, J. Mater. Process. Technol. 184 (2007) 115–123. https://doi.org/10.1016/j.jmatprotec.2006.11.024.

[29] H.S. Kim, M. Koç, Numerical investigations on springback characteristics of aluminum sheet metal alloys in warm forming conditions, J. Mater. Process. Technol. 204 (2008) 370–383. https://doi.org/10.1016/j.jmatprotec.2007.11.059.

[30] N. Abedrabbo, F. Pourboghrat, J. Carsley, Forming of aluminum alloys at elevated

temperatures – Part 1: Material characterization, Int. J. Plast. 22 (2006) 314–341. https://doi.org/10.1016/j.ijplas.2005.03.005.

[31] P.. Bolt, N.A.P.. Lamboo, P.J.C.. Rozier, Feasibility of warm drawing of aluminium products, J. Mater. Process. Technol. 115 (2001) 118–121. https://doi.org/10.1016/S0924-0136(01)00743-9.

[32] M. Kumar, C. Poletti, H.P. Degischer, Precipitation kinetics in warm forming of AW-7020 alloy, Mater. Sci. Eng. A. 561 (2013) 362–370. https://doi.org/10.1016/j.msea.2012.10.031.

[33] J. Lin, T.A. Dean, R.P. Garrett, A.D. Foster, Process for forming metal alloy sheet components, WO2008059242A2, 2008. https://patents.google.com/patent/WO2008059242A2/en.

[34] P.S. Raghupathi, M. Karima, N. Akgerman, T. Altan, A simplified approach to calculate springback in brake bending, in: Proc. NAMRC XI, 1983.

[35] Correcting springback. https://ahssinsights.org/forming/springback/correcting-springback/ (accessed July 20, 2022).

[36] X.A. Yang, F. Ruan, A die design method for springback compensation based on displacement adjustment, Int. J. Mech. Sci. 53 (2011) 399–406. https://doi.org/10.1016/j.ijmecsci.2011.03.002.

[37] Z. Zhang, J. Wu, S. Zhang, M. Wang, R. Guo, S. Guo, A new iterative method for springback control based on theory analysis and displacement adjustment, Int. J. Mech. Sci. 105 (2016) 330–339. https://doi.org/10.1016/j.ijmecsci.2015.11.005.

[38] R. Lingbeek, J. Huétink, S. Ohnimus, M. Petzoldt, J. Weiher, The development of a finite elements based springback compensation tool for sheet metal products, J. Mater. Process. Technol. 169 (2005) 115–125. https://doi.org/10.1016/j.jmatprotec.2005.04.027.

[39] H. Wang, J. Zhou, T. Zhao, Y. Tao, Springback compensation of automotive panel based on three-dimensional scanning and reverse engineering, Int. J. Adv. Manuf. Technol. 85 (2016) 1187–1193. https://doi.org/10.1007/s00170-015-8042-x.

[40] Y. Li, Q. Rong, Z. Shi, X. Sun, L. Meng, J. Lin, An accelerated springback compensation method for creep age forming, Int. J. Adv. Manuf. Technol. 102 (2019) 121–134. https://doi.org/10.1007/s00170-018-3175-3.

[41] K. Mori, M. Yamamoto, K. Osakada, Determination of hammering sequence in incremental sheet metal forming using a genetic algorithm, J. Mater. Process. Technol. 60 (1996) 463–468. https://doi.org/10.1016/0924-0136(96)02371-0.

[42] K.-I. Mori, Incremental hammering forming of sheet metal automated using CCD camera and database, Trans. NAMRI/SME. 26 (1998) 49–54. http://hdl.handle.net/10098/6427.

[43] H. Tanaka, N. Asakawa, M. Hirao, Development of a forging type rapid prototyping system; Automation of a free forging and metal hammering working, J. Robot. Mechatronics. 17 (2005) 523–528. https://doi.org/10.20965/jrm.2005.p0523.

[44] H. Iseki, K. Kato, S. Sakamoto, Flexible and incremental sheet metal forming using a spherical roller, in: Proc. 40th JJCTP, 1989: p. 44.

[45] H. Tanaka, N. Asakawa, M. Hirao, Forming type rapid prototyping development - error compensation with shape measurement -, Int. J. Autom. Technol. 2 (2008) 462–467. https://doi.org/10.20965/ijat.2008.p0462.

[46] N. Asakawa, H. Tanaka, Development of a forging-type rapid prototyping system (tool

path generation considering deformation process), Int. J. Autom. Technol. 4 (2010) 530–535. https://doi.org/10.20965/ijat.2010.p0530.

[47] K. Takasugi, H. Tanaka, M. Jono, N. Asakawa, Development of a forging type rapid prototyping system (relationship between hammering direction and product shape), Int. J. Autom. Technol. 6 (2012) 38–45. https://doi.org/10.20965/ijat.2012.p0038.

[48] H. Tanaka, K. Hoshino, N. Asakawa, Development of a CAM system with linear servo motor for automation of metal hammering, Key Eng. Mater. 625 (2014) 360–365. https://doi.org/10.4028/www.scientific.net/KEM.625.360.

[49] T. Schafer, R. Dieter Schraft, Incremental sheet metal forming by industrial robots, Rapid Prototyp. J. 11 (2005) 278–286. https://doi.org/10.1108/13552540510623585.

[50] Y. Luo, K. He, R. Du, A new sheet metal forming system based on incremental punching, part 2: machine building and experiment results, Int. J. Adv. Manuf. Technol. 51 (2010) 493–506. https://doi.org/10.1007/s00170-010-2635-1.

[51] J. Wang, L. Li, H. Jiang, Z. Zhang, S. Zhang, Incremental sheet punching on the base of sinusoidal tool path, Int. J. Adv. Manuf. Technol. 89 (2017) 3527–3534. https://doi.org/10.1007/s00170-016-9336-3.

[52] S.J. Yoon, D.Y. Yang, Investigation into a new incremental forming process using an adjustable punch set for the manufacture of a doubly curved sheet metal, Proc. Inst. Mech. Eng. Part B J. Eng. Manuf. 215 (2001) 991–1004. https://doi.org/10.1243/0954405011518791.

[53] B. Ilangovan, Fixtureless automated incremental sheet metal forming, Loughborough University, 2016.

[54] C.M. Bishop, Pattern recognition and machine learning, Springer, 2006.

[55] L. Monostori, A. Markus, H. Van Brussel, E. Westkämpfer, Machine learning approaches to manufacturing, CIRP Ann. 45 (1996) 675–712. https://doi.org/10.1016/s0007-8506(18)30216-6.

[56] I. Goodfellow, Y. Bengio, A. Courville, Deep learning, Cambridge, Massachusetts ; London, England : The MIT Press, 2016.

[57] D.P. Kingma, J.L. Ba, Adam: a method for stochastic optimization, 3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc. (2015) 1–15.

[58] J. Duchi, E. Hazan, Y. Singer, Adaptive subgradient methods for online learning and stochastic optimization, J. Mach. Learn. Res. 12 (2011) 2121–2159.

[59] G. Hinton, N. Srivastava, K. Swersky, Neural networks for machine learning - lecture 6a: overview of mini-batch gradient descent. https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf (accessed February 1, 2020).

[60] J.R. QUINLAN, Induction of decision trees, Mach. Learn. 1 (1986) 81–106.

[61] L. Breiman, Random forests, Mach. Learn. 45 (2001) pages5–32.

[62] M.A. Hearst, S.T. Dumais, E. Osuna, J. Platt, B. Scholkopf, Support vector machines, IEEE Intell. Syst. Their Appl. 13 (1998) 18–28. https://doi.org/10.1109/5254.708428.

[63] I. Rish, An empirical study of the naive Bayes classifier, Int. Jt. Conf. Artif. Intell. 2001 Work. Empir. Methods Artif. Intell. (2001) 41–46.

[64] N.S. Altman, An introduction to kernel and nearest-neighbor nonparametric regression, Am. Stat. 46 (1992) 175–185. https://doi.org/10.1080/00031305.1992.10475879.

[65] M. Hagan, H. Demuth, M. Beale, O. Jesús, Neuron model and network architectures, 2nd ed., 2014. http://hagan.okstate.edu/nnd.html.

[66] Q. V. Le, M. Ranzato, R. Monga, M. Devin, K. Chen, G.S. Corrado, J. Dean, A.Y. Ng, Building high-level features using large scale unsupervised learning, (2011). http://arxiv.org/abs/1112.6209.

[67] Y. Lecun, Y. Bengio, G. Hinton, Deep learning, Nature. 521 (2015) 436–444. https://doi.org/10.1038/nature14539.

[68] A.F. Agarap, Deep learning using rectified linear units (ReLU), (2018). http://arxiv.org/abs/1803.08375.

[69] S. Saha, A comprehensive guide to convolutional neural networks — the ELI5 way, (2018). https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 (accessed April 19, 2022).

[70] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks, Adv. Neural Inf. Process. Syst. 2 (2012) 1097–1105.

[71] J. Deng, W. Dong, R. Socher, L.-J. Li, Kai Li, Li Fei-Fei, ImageNet: A large-scale hierarchical image database, in: 2009 IEEE Conf. Comput. Vis. Pattern Recognit., IEEE, 2009: pp. 248–255. https://doi.org/10.1109/CVPR.2009.5206848.

[72] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (1997) 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735.

[73] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using RNN encoder-decoder for statistical machine translation, in: Proc. 2014 Conf. Empir. Methods Nat. Lang. Process., 2014: pp. 1724–1734. http://arxiv.org/abs/1406.1078.

[74] A.Y. Ng, Feature selection, L1 vs L2 regularization, and rotational invariance, in: 21 St Int. Conf. Mach. Learn., 2004.

[75] Y. Li, Research and application of deep learning in image recognition, in: 2022 IEEE 2nd Int. Conf. Power, Electron. Comput. Appl., IEEE, 2022: pp. 994–999. https://doi.org/10.1109/ICPECA53709.2022.9718847.

[76] P. Li, D. Wang, L. Wang, H. Lu, Deep visual tracking: Review and experimental comparison, Pattern Recognit. 76 (2018) 323–338. https://doi.org/10.1016/j.patcog.2017.11.007.

[77] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, P. Martinez-Gonzalez, J. Garcia-Rodriguez, A survey on deep learning techniques for image and video semantic segmentation, Appl. Soft Comput. 70 (2018) 41–65. https://doi.org/10.1016/j.asoc.2018.05.018.

[78] J. Su, B. Xu, H. Yin, A survey of deep learning approaches to image restoration, Neurocomputing. 487 (2022) 46–65. https://doi.org/10.1016/j.neucom.2022.02.046.

[79] K. Adnan, R. Akbar, Limitations of information extraction methods and techniques for heterogeneous unstructured big data, Int. J. Eng. Bus. Manag. 11 (2019) 184797901989077. https://doi.org/10.1177/1847979019890771.

[80] F. Stahlberg, Neural machine translation: A review, J. Artif. Intell. Res. 69 (2020) 343–418. https://doi.org/10.1613/jair.1.12007.

[81] P.K. Jain, R. Pamula, G. Srivastava, A systematic literature review on machine learning applications for consumer sentiment analysis using online reviews, Comput. Sci. Rev. 41 (2021) 100413. https://doi.org/10.1016/j.cosrev.2021.100413.

[82] R.M. Abdullah, S.Y. Ameen, D.M. Ahmed, S.F. Kak, H.M. Yasin, I.M. Ibrahim, A.M. Ahmed, Z.N. Rashid, N. Omar, A.A. Salih, Paralinguistic speech processing: An overview, Asian J. Res. Comput. Sci. (2021) 34–46. https://doi.org/10.9734/ajrcos/2021/v10i130233.

[83] J. Johnson, A. Karpathy, L. Fei-Fei, DenseCap: fully convolutional localization networks for dense captioning, Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. 2016-Decem (2016) 4565–4574. https://doi.org/10.1109/CVPR.2016.494.

[84] A.M. Tartakovsky, C.O. Marrero, P. Perdikaris, G.D. Tartakovsky, D. Barajas-Solano, Physics-informed deep neural networks for learning parameters and constitutive relationships in subsurface flow problems, Water Resour. Res. 56 (2020) 1–16. https://doi.org/10.1029/2019WR026731.

[85] R. Zhang, Y. Liu, H. Sun, Physics-informed multi-LSTM networks for metamodeling of nonlinear structures, Comput. Methods Appl. Mech. Eng. 369 (2020) 113226. https://doi.org/10.1016/j.cma.2020.113226.

[86] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707. https://doi.org/10.1016/j.jcp.2018.10.045.

[87] M.A. Nabian, H. Meidani, Physics-driven regularization of deep neural networks for enhanced engineering design and analysis, J. Comput. Inf. Sci. Eng. 20 (2020) 1–10. https://doi.org/10.1115/1.4044507.

[88] H. Yao, Y. Gao, Y. Liu, FEA-Net: A physics-guided data-driven model for efficient mechanical response prediction, Comput. Methods Appl. Mech. Eng. 363 (2020). https://doi.org/10.1016/j.cma.2020.112892.

[89] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, 3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc. (2015) 1–14.

[90] M. Lin, Q. Chen, S. Yan, Network in network, in: Proc. 2014 Int. Conf. Learn. Represent., Canada, 2013. http://arxiv.org/abs/1312.4400.

[91] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, (2015). http://arxiv.org/abs/1512.03385.

[92] G. Huang, Z. Liu, L. Van Der Maaten, K.Q. Weinberger, Densely connected convolutional networks, in: 2017 IEEE Conf. Comput. Vis. Pattern Recognit., IEEE, 2017: pp. 2261–2269. https://doi.org/10.1109/CVPR.2017.243.

[93] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W. Wong, W. Woo, Convolutional LSTM network: A machine learning approach for precipitation nowcasting, (2015). http://arxiv.org/abs/1506.04214.

[94] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, Y. Bengio, Show, attend and tell: Neural image caption generation with visual attention, in: Proc. 32nd Int. Conf. Mach. Learn., Lille, France, 2015: pp. 2048–2057.

[95] R.S. Sutton, A.G. Barto, Reinforcement learning: An introduction, Second edi, The MIT Press Cambridge, Massachusetts London, England, 2017.

[96] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, D. Hassabis, Human-level

control through deep reinforcement learning, Nature. 518 (2015) 529–533. https://doi.org/10.1038/nature14236.

[97] H. van Hasselt, A. Guez, D. Silver, Deep reinforcement learning with double Q-learning, (2015). http://arxiv.org/abs/1509.06461.

[98] Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, N. de Freitas, Dueling network architectures for deep reinforcement learning, (2015). http://arxiv.org/abs/1511.06581.

[99] V. Mnih, A.P. Badia, M. Mirza, A. Graves, T.P. Lillicrap, T. Harley, D. Silver, K. Kavukcuoglu, Asynchronous methods for deep reinforcement learning, (2016). http://arxiv.org/abs/1602.01783.

[100] T.P. Lillicrap, J.J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning, (2015). http://arxiv.org/abs/1509.02971.

[101] J. Schulman, S. Levine, P. Moritz, M.I. Jordan, P. Abbeel, Trust region policy optimization, (2015). http://arxiv.org/abs/1502.05477.

[102] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, (2017). http://arxiv.org/abs/1707.06347.

[103] J. Schulman, P. Moritz, S. Levine, M. Jordan, P. Abbeel, High-dimensional continuous control using generalized advantage estimation, (2015). http://arxiv.org/abs/1506.02438.

[104] K. Arulkumaran, M.P. Deisenroth, M. Brundage, A.A. Bharath, Deep reinforcement learning: A brief survey, IEEE Signal Process. Mag. 34 (2017) 26–38. https://doi.org/10.1109/MSP.2017.2743240.

[105] M. Kempka, M. Wydmuch, G. Runc, J. Toczek, W. Jaskowski, ViZDoom: A Doom-based AI research platform for visual reinforcement learning, in: 2016 IEEE Conf. Comput. Intell. Games, IEEE, 2016: pp. 1–8. https://doi.org/10.1109/CIG.2016.7860433.

[106] O. Vinyals, I. Babuschkin, W.M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D.H. Choi, R. Powell, T. Ewalds, P. Georgiev, J. Oh, D. Horgan, M. Kroiss, I. Danihelka, A. Huang, L. Sifre, T. Cai, J.P. Agapiou, M. Jaderberg, A.S. Vezhnevets, R. Leblond, T. Pohlen, V. Dalibard, D. Budden, Y. Sulsky, J. Molloy, T.L. Paine, C. Gulcehre, Z. Wang, T. Pfaff, Y. Wu, R. Ring, D. Yogatama, D. Wünsch, K. McKinney, O. Smith, T. Schaul, T. Lillicrap, K. Kavukcuoglu, D. Hassabis, C. Apps, D. Silver, Grandmaster level in StarCraft II using multi-agent reinforcement learning, Nature. 575 (2019) 350–354. https://doi.org/10.1038/s41586-019-1724-z.

[107] D. Silver, A. Huang, C.J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, D. Hassabis, Mastering the game of go with deep neural networks and tree search, Nature. 529 (2016) 484–489. https://doi.org/10.1038/nature16961.

[108] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, D. Hassabis, Mastering the game of Go without human knowledge, Nature. 550 (2017) 354–359. https://doi.org/10.1038/nature24270.

[109] J. Heinrich, D. Silver, Deep reinforcement learning from self-play in imperfect-information games, in: NIPS 2016 Deep Reinf. Learn. Work., 2016. http://arxiv.org/abs/1603.01121.

[110] M. Moravčík, M. Schmid, N. Burch, V. Lisý, D. Morrill, N. Bard, T. Davis, K. Waugh, M. Johanson, M. Bowling, DeepStack: Expert-level artificial intelligence in heads-up no-limit poker, Science. 356 (2017) 508–513. https://doi.org/10.1126/science.aam6960.

[111] T. Haarnoja, V. Pong, A. Zhou, M. Dalal, P. Abbeel, S. Levine, Composable deep reinforcement learning for robotic manipulation, in: 2018 IEEE Int. Conf. Robot. Autom., IEEE, 2018: pp. 6244–6251. https://doi.org/10.1109/ICRA.2018.8460756.

[112] O.M. Andrychowicz, B. Baker, M. Chociej, R. Józefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, W. Zaremba, Learning dexterous in-hand manipulation, Int. J. Rob. Res. 39 (2020) 3–20. https://doi.org/10.1177/0278364919887447.

[113] Y. Zhu, R. Mottaghi, E. Kolve, J.J. Lim, A. Gupta, L. Fei-Fei, A. Farhadi, Target-driven visual navigation in indoor scenes using deep reinforcement learning, in: 2017 IEEE Int. Conf. Robot. Autom., IEEE, 2017: pp. 3357–3364. https://doi.org/10.1109/ICRA.2017.7989381.

[114] P. Shah, M. Fiser, A. Faust, J.C. Kew, D. Hakkani-Tur, FollowNet: Robot navigation by following natural language directions with deep reinforcement learning, (2018). http://arxiv.org/abs/1805.06150.

[115] C. García, Artificial intelligence applied to automatic supervision, diagnosis and control in sheet metal stamping processes, J. Mater. Process. Technol. 164–165 (2005) 1351–1357. https://doi.org/10.1016/j.jmatprotec.2005.02.031.

[116] J. Zhao, F. Wang, Parameter identification by neural network for intelligent deep drawing of axisymmetric workpieces, J. Mater. Process. Technol. 166 (2005) 387–391. https://doi.org/10.1016/j.jmatprotec.2004.08.020.

[117] M.A. Dib, N.J. Oliveira, A.E. Marques, M.C. Oliveira, J. V. Fernandes, B.M. Ribeiro, P.A. Prates, Single and ensemble classifiers for defect prediction in sheet metal forming under variability, Neural Comput. Appl. 32 (2020) 12335–12349. https://doi.org/10.1007/s00521-019-04651-6.

[118] T. Tsuruya, M. Danseko, K. Sasaki, S. Honda, R. Takeda, Forming state recognition in deep drawing process with machine learning, J. Adv. Mech. Des. Syst. Manuf. 13 (2019) JAMDSM0066–JAMDSM0066. https://doi.org/10.1299/jamdsm.2019jamdsm0066.

[119] S.B. Block, R.D. da Silva, L.B. Dorini, R. Minetto, Inspection of imprint defects in stamped metal surfaces using deep learning and tracking, IEEE Trans. Ind. Electron. 68 (2021) 4498–4507. https://doi.org/10.1109/TIE.2020.2984453.

[120] C.Y. Huang, Z. Dzulfikri, Stamping monitoring by using an adaptive 1D convolutional neural network, Sensors (Switzerland). 21 (2021) 1–21. https://doi.org/10.3390/s21010262.

[121] D.A. Molitor, C. Kubik, M. Becker, R.H. Hetfleisch, F. Lyu, P. Groche, Towards high-performance deep learning models in tool wear classification with generative adversarial networks, J. Mater. Process. Technol. 302 (2022) 117484. https://doi.org/10.1016/j.jmatprotec.2021.117484.

[122] J.M. Griffin, V. V. Shanbhag, M.P. Pereira, B.F. Rolfe, Application of machine learning for acoustic emissions waveform to classify galling wear on sheet metal stamping tools, Int. J. Adv. Manuf. Technol. 116 (2021) 579–596. https://doi.org/10.1007/s00170-021-07408-5.

[123] D.A. Molitor, C. Kubik, R.H. Hetfleisch, P. Groche, Workpiece image-based tool wear

classification in blanking processes using deep convolutional neural networks, (2021). http://arxiv.org/abs/2107.12034.

[124] S. Kashid, S. Kumar, Prediction of life of punches of compound die using artificial neural network, in: 5th Int. 26th All India Manuf. Technol. Des. Res. Conf. (AIMTDR 2014), IIT Guwahati, Assam, India, 2014: pp. 62-1-62–6.

[125] P.-J. Chen, T.-L. Wu, T.-J. Lin, Y.-H. Lai, T.-L. Wu, C. Yeh, Life period estimation of stamping process using punch sounds and deep neural network, in: 2019 14th IEEE Conf. Ind. Electron. Appl., IEEE, 2019: pp. 625–628. https://doi.org/10.1109/ICIEA.2019.8833634.

[126] E. Lejon, P. Kyösti, J. Lindström, Machine learning for detection of anomalies in press-hardening: Selection of efficient methods, in: Procedia CIRP, 2018: pp. 1079–1083. https://doi.org/10.1016/j.procir.2018.03.221.

[127] G. Zhang, C. Li, H. Zhou, T. Wagner, Punching process monitoring using wavelet transform based feature extraction and semi-supervised clustering, Procedia Manuf. 26 (2018) 1204–1212. https://doi.org/10.1016/j.promfg.2018.07.156.

[128] M.. Inamdar, P.. Date, U.. Desai, Studies on the prediction of springback in air vee bending of metallic sheets using an artificial neural network, J. Mater. Process. Technol. 108 (2000) 45–54. https://doi.org/10.1016/S0924-0136(00)00588-4.

[129] R. Narayanasamy, P. Padmanabhan, Comparison of regression and artificial neural network model for the prediction of springback during air bending process of interstitial free steel sheet, J. Intell. Manuf. 23 (2012) 357–364. https://doi.org/10.1007/s10845-009-0375-6.

[130] W. Liu, Q. Liu, F. Ruan, Z. Liang, H. Qiu, Springback prediction for sheet metal forming based on GA-ANN technology, J. Mater. Process. Technol. 187–188 (2007) 227–231. https://doi.org/10.1016/j.jmatprotec.2006.11.087.

[131] Z. Guo, W. Tang, Bending angle prediction model based on BPNN-spline in air bending springback process, Math. Probl. Eng. 2017 (2017) 1–11. https://doi.org/10.1155/2017/7834621.

[132] S.C. Spathopoulos, G.E. Stavroulakis, Springback prediction in sheet metal forming, based on finite element analysis and artificial neural network approach, Appl. Mech. 1 (2020) 97–110. https://doi.org/10.3390/applmech1020007.

[133] S.K. Singh, D.R. Kumar, Application of a neural network to predict thickness strains and finite element simulation of hydro-mechanical deep drawing, Int. J. Adv. Manuf. Technol. 25 (2005) 101–107. https://doi.org/10.1007/s00170-003-1842-4.

[134] K. Elangovan, C. Sathiya Narayanan, R. Narayanasamy, Modelling of forming limit diagram of perforated commercial pure aluminium sheets using artificial neural network, Comput. Mater. Sci. 47 (2010) 1072–1078. https://doi.org/10.1016/j.commatsci.2009.12.016.

[135] H. Cai, W. Xiao, K. Zheng, The prediction of part thickness using machine learning in aluminum hot stamping process with partition temperature control, Int. J. Adv. Manuf. Technol. 119 (2022) 3891–3902. https://doi.org/10.1007/s00170-021-08632-9.

[136] T. Hart-Rawung, J. Buhl, M. Bambach, A fast approach for optimization of hot stamping based on machine learning of phase transformation kinetics, Procedia Manuf. 47 (2020) 707–712. https://doi.org/10.1016/j.promfg.2020.04.218.

[137] H.R. Attar, H. Zhou, A. Foster, N. Li, Rapid feasibility assessment of components to be

formed through hot stamping: A deep learning approach, J. Manuf. Process. 68 (2021) 1650–1671. https://doi.org/10.1016/j.jmapro.2021.06.011.

[138] H. Zhou, Q. Xu, Z. Nie, N. Li, A study on using image-based machine learning methods to develop surrogate models of stamp forming simulations, J. Manuf. Sci. Eng. 144 (2022). https://doi.org/10.1115/1.4051604.

[139] R. Roy, Assessment of sheet-metal bending requirements using neural networks, Neural Comput. Appl. 4 (1996) 35–43. https://doi.org/10.1007/BF01413868.

[140] N. Ivezic, J.D. Allen Jr., T. Zacharia, Neural network method for inverse modeling of material deformation, in: Second Int. Conferemce Intell. Process. Manuf. Mater. IPPM'99, 1999: pp. 961–966.

[141] E. Hamouche, E.G. Loukaides, Classification and selection of sheet forming processes with machine learning, Int. J. Comput. Integr. Manuf. 31 (2018) 921–932. https://doi.org/10.1080/0951192X.2018.1429668.

[142] D. Opritescu, W. Volk, Automated driving for individualized sheet metal part production - A neural network approach, Robot. Comput. Integr. Manuf. 35 (2015) 144–150. https://doi.org/10.1016/j.rcim.2015.03.006.

[143] C. Hartmann, D. Opritescu, W. Volk, An artificial neural network approach for tool path generation in incremental sheet metal free-forming, J. Intell. Manuf. 30 (2019) 757–770. https://doi.org/10.1007/s10845-016-1279-x.

[144] G. Rossi, P. Nicholas, Re/Learning the wheel: Methods to utilize neural networks as design tools for doubly curved metal surfaces, in: Proc. 38th Annu. Conf. Assoc. Comput. Aided Des. Archit., Mexico City, 2018: pp. 146–155.

[145] D.D. Störkle, P. Seim, L. Thyssen, B. Kuhlenkötter, Machine learning in incremental sheet forming, in: 47st Int. Symp. Robot., 2016: pp. 1–7.

[146] D. Störkle, P. Altmann, D. Möllensiep, L. Thyssen, B. Kuhlenkötter, Automated parameterization of local support at every toolpath point in robot-based incremental sheet forming, in: Procedia Manuf., 2019: pp. 67–73. https://doi.org/10.1016/j.promfg.2019.02.107.

[147] W.L. Xu, C.H. Ma, C.H. Li, W.J. Feng, Sensitive factors in springback simulation for sheet metal forming, J. Mater. Process. Technol. 151 (2004) 217–222. https://doi.org/10.1016/j.jmatprotec.2004.04.044.

[148] K.P. Li, W.P. Carden, R.H. Wagoner, Simulation of springback, Int. J. Mech. Sci. 44 (2002) 103–122. https://doi.org/10.1016/S0020-7403(01)00083-2.

[149] S. Liu, Y. Xia, Z. Shi, H. Yu, Z. Li, J. Lin, Deep learning in sheet metal bending with a novel theory-guided deep neural network, IEEE/CAA J. Autom. Sin. 8 (2021) 565–581. https://doi.org/10.1109/JAS.2021.1003871.

[150] Q. Rong, Y. Li, Z. Shi, L. Meng, X. Sun, X. Sun, J. Lin, Experimental investigations of stress-relaxation ageing behaviour of AA6082, Mater. Sci. Eng. A. 750 (2019) 108–116. https://doi.org/10.1016/j.msea.2019.02.043.

[151] H. Sun, X. Song, R. Ma, J. Zhao, Theoretical analysis of JCO forming and springback for sheet metal four-point bending in manufacture of LSAW pipe, J. Mech. Eng. 25 (2014) 257–262.

[152] Z. Marciniak, J.L. Duncan, S.J. Hu, Mechanics of sheet metal forming, Elsevier, 2002. https://doi.org/10.1016/B978-0-7506-5300-8.X5000-6.

[153] C. Zimmerling, D. Trippe, B. Fengler, L. Kärger, An approach for rapid prediction of

textile draping results for variable composite component geometries using deep neural networks, AIP Conf. Proc. 2113 (2019). https://doi.org/10.1063/1.5112512.

[154] R. Caruana, Multitask connectionist learning, in: Proc. 1993 Connect. Model. Summer Sch., 1993: pp. 372–379.

[155] R. Ranjan, V.M. Patel, R. Chellappa, HyperFace: a deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition, IEEE Trans. Pattern Anal. Mach. Intell. 41 (2019) 121–135. https://doi.org/10.1109/TPAMI.2017.2781233.

[156] R. Collobert, J. Weston, A unified architecture for natural language processing: deep neural networks with multitask learning, Proc. 25th Int. Conf. Mach. Learn. (2008) 160–167.

[157] M.L. Seltzer, J. Droppo, Multi-task learning in deep neural networks for improved phoneme recognition, ICASSP, IEEE Int. Conf. Acoust. Speech Signal Process. - Proc. (2013) 6965–6969. https://doi.org/10.1109/ICASSP.2013.6639012.

[158] S. Liu, Y. Xia, Y. Liu, Z. Shi, H. Yu, Z. Li, J. Lin, Tool path planning of consecutive free-form sheet metal stamping with deep learning, J. Mater. Process. Technol. 303 (2022) 117530. https://doi.org/10.1016/j.jmatprotec.2022.117530.

[159] Material contact properties table. http://atc.sjf.stuba.sk/files/mechanika_vms_ADAMS/Contact_Table.pdf (accessed October 4, 2022).

[160] Cambridge University Engineering Department, Materials data book, (2003). http://www-mdp.eng.cam.ac.uk/web/library/enginfo/cueddatabooks/materials.pdf (accessed February 13, 2020).

[161] E. Shelhamer, J. Long, T. Darrell, Fully convolutional networks for semantic segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 39 (2017) 640–651. https://doi.org/10.1109/TPAMI.2016.2572683.

[162] O. Ronneberger, P. Fischer, T. Brox, U-Net: Convolutional networks for biomedical image segmentation, ArXiv. (2015).

[163] Z. Nie, H. Jiang, L.B. Kara, Stress field prediction in cantilevered structures using convolutional neural networks, J. Comput. Inf. Sci. Eng. 20 (2020). https://doi.org/10.1115/1.4044097.

[164] P. Isola, J.Y. Zhu, T. Zhou, A.A. Efros, Image-to-image translation with conditional adversarial networks, Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017. 2017-Janua (2017) 5967–5976. https://doi.org/10.1109/CVPR.2017.632.

[165] O. Vinyals, A. Toshev, S. Bengio, D. Erhan, Show and tell: A neural image caption generator, Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. 07-12-June (2015) 3156–3164. https://doi.org/10.1109/CVPR.2015.7298935.

[166] L. van der Maaten, G. Hinton, Visualizing high-dimensional data using t-SNE, J. Mach. Learn. Res. 9 (2008) 2579–2605.

[167] K. Enomoto, K. Sakurada, W. Wang, H. Fukui, M. Matsuoka, R. Nakamura, N. Kawaguchi, Filmy cloud removal on satellite imagery with multispectral conditional generative adversarial nets, in: 2017 IEEE Conf. Comput. Vis. Pattern Recognit. Work., IEEE, 2017: pp. 1533–1541. https://doi.org/10.1109/CVPRW.2017.197.

[168] Y. Xia, H. Yu, F.-Y. Wang, Accurate and robust eye center localization via fully convolutional networks, IEEE/CAA J. Autom. Sin. 6 (2019) 1127–1138.

https://doi.org/10.1109/JAS.2019.1911684.

[169] H. Lee, J. Lee, H. Kim, B. Cho, S. Cho, Deep-neural-network-based sinogram synthesis for sparse-view CT image reconstruction, IEEE Trans. Radiat. Plasma Med. Sci. 3 (2019) 109–119. https://doi.org/10.1109/TRPMS.2018.2867611.

[170] S. Liu, Z. Shi, J. Lin, Z. Li, Reinforcement learning in free-form stamping of sheet-metals, Procedia Manuf. 50 (2020) 444–449. https://doi.org/10.1016/j.promfg.2020.08.081.

[171] Aluminium alloys - aluminium 1050 properties, fabrication and applications. https://www.azom.com/article.aspx?ArticleID=2798 (accessed July 19, 2022).

[172] R. Kirk, A. Zhang, E. Grefenstette, T. Rocktäschel, A survey of generalisation in deep reinforcement learning, (2021). http://arxiv.org/abs/2111.09794.

[173] G. Matheron, N. Perrin, O. Sigaud, The problem with DDPG: understanding failures in deterministic environments with sparse rewards, (2019). http://arxiv.org/abs/1911.11679.

[174] R.J. Williams, J. Peng, Function optimization using connectionist reinforcement learning algorithms, Conn. Sci. 3 (1991) 241–268. https://doi.org/10.1080/09540099108946587.

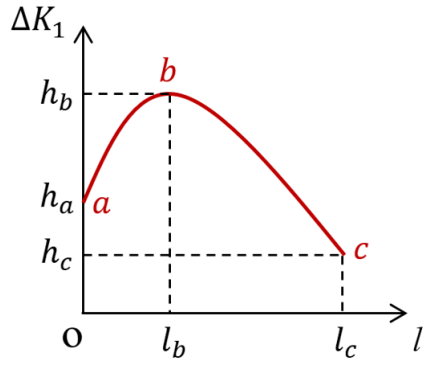
Appendix A. Virtual environment configuration

The virtual environment (VE) is developed to imitate the rubber-tool forming behaviours in FE simulations, in which the DRL algorithms are trained to reduce computational expense. Since the forming process is extremely nonlinear, VE is only tweaked to qualitatively resemble FE simulation results, which, however, is sufficient for RL algorithms comparison. The VE is configured following the rules below:

1. A single punch operation only affects the $\Delta\boldsymbol{K}$ values at 50 node locations (5 mm) around the punch location and the punch location itself in the $\Delta\boldsymbol{K}$-graph.
2. If a node location has been punched with a stroke and this location is selected again for punching, the $\Delta\boldsymbol{K}$-graph will only change if the new stroke is greater than the previous one.
3. The change of $\boldsymbol{K}$ value (c_K) at the punch location by stroke (d_s) without CPE1 and CPE2 is defined as: $c_{K0} = (d_s - 2.1) \times 0.05 + 0.045$.
4. With CPE1:
    i. only one side of the punch location is pre-deformed: $c_{K1} = c_{K0}/2$;
    ii. both sides of the punch location are pre-deformed by 1 punch: $c_{K2} = (c_{K0} + 0.035)/2$;
    iii. one side of the punch location is pre-deformed by 1 punch and the other side is pre-deformed by 2 punches: $c_{K3} = (c_{K0} + 0.005)/2$;
    iv. two sides of the punch locations are pre-deformed by 4 and over 4 punches in total: $c_{K4} = (c_{K0} - 0.01)/2$.
5. The change of $\boldsymbol{K}$ value gradually decreases from c_{Ki} ($i \in [0,4], \mathbb{Z}$) at the punch location to 0 at two ends of the 51 node locations in rule 1.
6. With CPE2: if the $\Delta\boldsymbol{K}$-graph is changed by the punch, $\Delta\boldsymbol{K}$ values at the 51 node locations in rule 1 are reduced by 0.0005 mm^{-1}.

Appendix B. Arbitrary generation of workpiece segments

The $\Delta \boldsymbol{K}$-graph of each variant in Group 1 shown in Figure B1, which is composed of two parabolas $\Delta \boldsymbol{K}_{a-b}$ and $\Delta \boldsymbol{K}_{b-c}$, is determined by five variables (h_a, h_c, l_b, w_{ab} and w_{bc}) and two constants (h_b and l_c).



- h_b: 0.05 mm^{-1}
- h_a: 0 - 0.005 mm^{-1}
- h_c: 0 - 0.005 mm^{-1}
- l_b: 5 - 15 mm
- l_c: 30 mm
- $\Delta K_{a-b} = w_{ab}(l - l_b)^2 + h_b$
- $\Delta K_{b-c} = w_{bc}(l - l_b)^2 + h_b$

Figure B1. The variables and functions for creating the variants of segments in Group 1. w_{ab} and w_{bc} can be derived once h_a, h_b, h_c and l_b are generated.

Appendix C. Extraordinarily large CPE2

A phenomenon of extraordinarily large CPE2 is shown in Figure C1. It can be seen that, after a punch with stroke of 3.6 mm was applied to location 76, there is an evident effect of curvature at about location 70. When the applied stroke was increased by 0.1 mm, the CPE2 at location 70 increased by about 0.007 mm^{-1}.

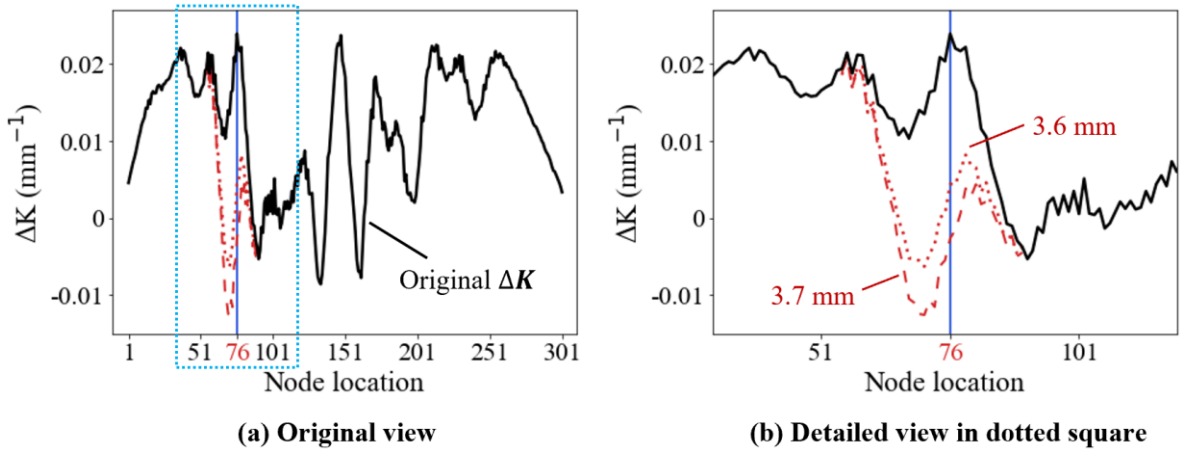


Figure C1. Extraordinarily large CPE2 occurring near punch location of 80. The vertical line denotes the punch location at the current workpiece state (original ΔK-graph). The dotted line and dashed line denote the ΔK-graphs after punches with stroke of 3.6 mm and 3.7 mm, respectively.