



A generalisable tool path planning strategy for free-form sheet metal stamping through deep reinforcement and supervised learning

Shiming Liu¹ · Zhusheng Shi¹ · Jianguo Lin¹ · Hui Yu²

Received: 8 November 2022 / Accepted: 13 March 2024
© The Author(s) 2024

Abstract

Due to the high cost of specially customised presses and dies and the advance of machine learning technology, there is some emerging research attempting free-form sheet metal stamping processes which use several common tools to produce products of various shapes. However, tool path planning strategies for the free forming process, such as reinforcement learning technique, derived from previous path planning experience are not generalisable for an arbitrary new sheet metal workpiece. Thus, in this paper, a generalisable tool path planning strategy is proposed for the first time to realise the tool path prediction for an arbitrary sheet metal part in 2-D space with no metal forming knowledge in prior, through deep reinforcement (implemented with 2 heuristics) and supervised learning technologies. Conferred by deep learning, the tool path planning process is corroborated to have self-learning characteristics. This method has been instantiated and verified by a successful application to a case study, of which the workpiece shape deformed by the predicted tool path has been compared with its target shape. The proposed method significantly improves the generalisation of tool path planning of free-form sheet metal stamping process, compared to strategies using pure reinforcement learning technologies. The successful instantiation of this method also implies the potential of the development of intelligent free-form sheet metal stamping process.

Keywords Deep learning · Deep reinforcement learning · Deep supervised learning · Sheet metal forming · Intelligent manufacturing · Tool path planning

Introduction

Sheet metal components are nowadays ubiquitous in various industrial products, such as automobile, aircrafts and high-speed trains. 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. However, the formed products from sheet metal stamping technology are subject to the unalterable shapes of punch and dies, for which the limited forming flexibility impedes the applicability of the off-the-shelf stamping equipment

to new sheet metal components. In addition, the extraordinarily high capital cost for specialised punches and dies, especially for large-scale stamping, leads to expensive prototyping and arduous research and development of novel sheet metal designs. Thus, to extricate sheet metal manufacture from these constraints and to fulfill the requirement of high-volume personalised production in sheet metal forming industry nowadays (Bowen et al., 2022), flexible forming processes, which can change workpiece geometry without requiring different tool sets, were developed (Allwood & Utsunomiya, 2006). An emerging free-form sheet metal stamping technique was brought up (Liu et al., 2022), which consecutively deforms a sheet metal to its target shape from blank using several small-scale punch and dies of different shapes. In this regard, of particular concern is the generation and optimisation of the forming tool path which could yield the forming result comparable to the forming target.

Due to the forming characteristics of the traditional sheet metal stamping process, the sheet metal part is usually formed within a few or just one forming step, for which no research

✉ Zhusheng Shi
zhusheng.shi@imperial.ac.uk

¹ Department of Mechanical Engineering, Imperial College London, London SW7 2AZ, UK

² School of Creative Technologies, University of Portsmouth, Portsmouth PO1 2DJ, UK

on tool path for stamping can be found. In sheet metal forming industry, most studies involving tool path generation and optimisation were performed for incremental sheet metal forming (ISF) process, which deforms sheet metal to its target shape with a sequence of incremental deformations. Attanasio et al. (2006) manually designed several tool paths for a two point ISF to manufacture an automotive part, by varying the step depth and scallop height. They found that setting low values of both these parameters can improve the final dimensional accuracy and surface quality. Similarly, Tanaka et al. (2005) manually generated tool paths for an incremental sheet punching (ISP) process based on the target workpiece CAD, tool shape, crossfeed, depth and tool path mode, of which the deformed workpiece had a maximum length of 76 mm. Azaouzi and Lebaal (2012) proposed a tool path optimisation strategy for single point ISF using the response surface method and sequential quadratic programming algorithm, which was tested for a spiral tool path and realised through finite element analysis (FEA). This method was reported to reduce the manufacturing time and improve the homogeneity of thickness distribution of asymmetric parts. Malhotra et al. (2011) proposed a tool path generation strategy to alleviate the unintentionally formed stepped features on the component base occurring in a multi-pass single point ISF process, by combining in-to-out and out-to-in tool paths for each intermediate shape. It was found that this strategy effectively reduced the occurrence of stepped features compared to pure out-to-in tool paths.

Over the past decade, machine learning technology has seen its unprecedented development in image recognition and natural language processing thanks to the remarkably increased computation power of central processing units (CPUs). Impressed by its extraordinary learning capability, researchers started to harness machine learning or deep learning technologies in sheet metal forming industry, such as ISF (Nagargoje et al., 2021). Most of them focused on process monitoring (Kubik et al., 2022), surrogate model for forming results prediction (Low et al., 2022) and process parameters prediction (Liu et al., 2021). Machine learning is well known through three categories of techniques (Monostori et al., 1996): supervised learning (SL), unsupervised learning, and reinforcement learning (RL). With regard to forming tool path planning, most applications exploited supervised and reinforcement learning techniques. Opritescu and Volk (2015) and Hartmann et al. (2019) utilised supervised learning neural networks for optimal tool path prediction for 2-D and 3-D automated driving processes (Kraftforming), respectively. Curvature distribution on target workpiece surface was computed as inputs, and they reported that the careful workpiece digitisation was of great importance to achieve good learning efficiency. The tool path for automated wheeling process was predicted by Rossi and Nicholas (2018) using fully convolutional network (FCN), with 75% prediction

accuracy. Störkle et al. (2019) used linear regressor, decision tree, random decision forest, support vector machine and Gaussian process regressor to predict the optimal local support force and support angle distribution along a tool path in an ISF process. Liu et al. (2022) developed a recursive tool path prediction framework for a rubber-tool forming process, which embedded a deep supervised learning model for tool path planning. They compared the performance of three series of state-of-the-art models, including single CNNs, cascaded networks and convolutional long short-term memory (LSTM) models in tool path learning, from which the convolutional LSTM was reported to be the most superior. Compared to supervised learning, reinforcement learning applications to tool path planning of sheet metal forming process have been significantly ignored. This could be due to the expensive acquisition of computational or experimental data for RL algorithms training. Störkle et al. (2016) proposed a RL-based approach for the tool path planning and adjustment of an ISF process, which increased the geometric accuracy of the formed part. Liu et al. (2020) used a reinforcement learning algorithm, namely deep Q-learning, for the tool path learning of a simple free-form sheet metal stamping process. The FE computation was interfaced to the Q-learning algorithm as the RL environment, which provided real-time forming data for algorithm training.

Although there have been numerous studies of tool path planning for various sheet metal forming processes, they all have a common issue in generalising the methods to completely different target workpiece shapes, which hinders the widespread applications of machine learning based tool path planning strategies. In other words, new data have to be acquired to train the machine learning models or algorithms again to have a good prediction accuracy for different target, especially for approaches exploiting reinforcement learning. Generalisation gap is a common issue in RL applications (Kirk et al., 2021), 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.

Table 1 briefly compares the methods introduced above in tool path planning and summarises their deficiency in terms of real-world application. “Curse of dimensionality” indicates that the method can be error-prone once the target workpiece shape becomes complex, since the available data would become sparse and exponentially increased training data is required to obtain a reliable prediction result.

The aim of this research is to explore the generalisation of deep learning technologies in forming tool path planning for a 2-D free-form sheet metal stamping process. A generalisable tool path planning strategy, through the design

Table 1 Comparison between different tool path planning methods/algorithms

References	Tool path planning methods/algorithms description	Deficiency
Attanasio et al. (2006) Tanaka et al. (2005)	Manually generate tool path for ISF and ISP process	Labour intensive
Azaouzi and Lebaal (2012)	Use optimisation algorithm to iteratively solve the optimal tool path	Not generalisable
Malhotra et al. (2011)	Improve tool path through analytical modelling and experimental analysis	Not generalisable
Opritescu and Volk (2015) Hartmann et al. (2019)	Use neural network to predict tool path for a Kraftforming process	“Curse of dimensionality”
Rossi and Nicholas (2018)	Use FCN to predict tool path for an automated wheeling process	“Curse of dimensionality”
Störkle et al. (2019)	Use traditional machine learning to predict forming parameter sequence in an ISF process	“Curse of dimensionality”
Liu et al. (2022)	Use deep neural networks to predict tool path for a rubber-tool forming process	“Curse of dimensionality”
Störkle et al. (2016)	Use RL to optimise the tool path for an ISF process	Not generalisable
Liu et al. (2020)	Use RL to learn the optimal tool path for a free-form sheet metal stamping process	Not generalisable

of deep reinforcement and deep supervised learning technologies at different stages, was proposed in this paper. In this strategy, RL was used to explore the optimal tool paths for the target workpiece, with which the efficient tool path for a certain group of workpieces was learned using SL. With no metal forming knowledge in prior, the path planning process was corroborated to possess self-learning characteristics, from which the path planning results can be self-improved over time. The generalisation of this strategy was realised by factorising the entire target workpiece into several segments, which were classified into three groups. The optimal tool paths for several typical workpiece segments from each group were learned from scratch through deep reinforcement learning, and deep supervised learning

models were used to generalise the intrinsic forming pattern of each group of segments. Six deep RL algorithms, from two different categories, were compared regarding their tool path learning performance for the free-form stamping process. The RL process was enhanced with the introduction of two forming heuristics. Three deep SL models were trained with two tool path datasets consist of different data amount and their performance were evaluated in terms of forming goal achievement and the dimension error of the deformed workpiece, and the forming results from a pure reinforcement learning method were also presented as comparisons. At last, a case study was performed to verify the generalisable tool path planning strategy with a completely new target workpiece.

The main contributions of this work are as following: 1) developing a generalisable tool path planning strategy for arbitrary 2-D free-formed sheet metal components for the first time, which successfully integrated deep RL and SL algorithms to learn and generalise efficient forming paths, and validating through a case study; 2) analysing a free-form rubber-tool forming process and discovering 2 close punch effects; 3) quantitatively analysing the performance of 6 deep RL algorithms and 3 deep SL models on tool path learning and generalisation, respectively. In addition, two heuristics were derived from real-world empirical experience and have been demonstrated to significantly facilitate the tool path learning process.

Methodology

In this section, the application of the proposed tool path planning strategy was first introduced in "[Free-form stamping test and digitisation of forming process](#)" section, followed by the detailed illustration of the generalisable tool path planning strategy in "[Generalisable tool path planning strategy](#)" section. "[Forming goal and forming parameters design](#)" section presents the forming goal that the strategy needs to achieve and the forming parameters to be selected. "[Deep reinforcement learning algorithms and learning parameters](#)" section and 2.5 illustrate the designation details of the RL and SL algorithms, respectively.

Free-form stamping test and digitisation of forming process

A rubber-tool forming process proposed in the Authors' previous research (Liu et al., 2022) 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 Fig. 1, the workpiece was deformed by a rubber-wrapped punch on a workbench rubber. The specification of the setup is summarised in Table 2. The deformation

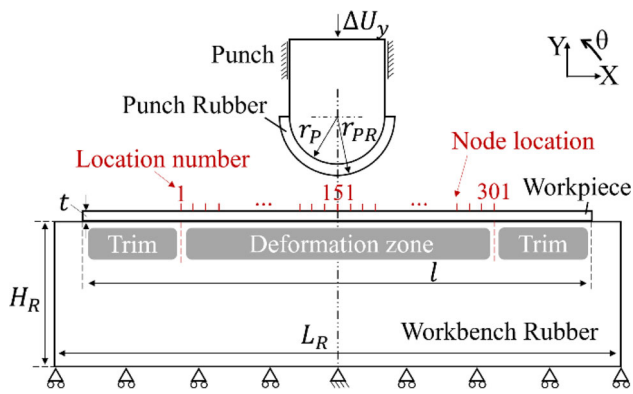


Fig. 1 Test setup and FE model for the rubber-tool forming. The lengths of the deformation and trim zones are 30 and 10 mm, respectively

was accomplished by translating the punch towards the workpiece along Y-axis and lifting it up, considering springback. The workpiece was consecutively deformed at different locations towards its target shape. At each step of the free forming process, the workpiece was repositioned through rotation and translation to relocate the punch location, the details of which can be found in (Liu et al., 2022). 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, with details in (Liu et al., 2022). Mesh for the workpiece was of size 0.1 mm, with 17,164 elements in total.

To realise the free forming process in FE simulations, the forming process was digitised and standardised for precise process control. As shown in Fig. 1, the workpiece was divided into two zones, namely deformation zone and trim zone, with lengths of 30 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 are 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 Fig. 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 K -graph.

Generalisable tool path planning strategy

The proposed generalisable tool path planning 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. From the theoretical perspective, 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. From the empirical perspective, representative groups of workpiece segments are finite, while there are infinite number of possible target workpiece shapes. After studying the efficient forming path for each segment group, the tool path for any arbitrary workpiece can be obtained by aggregating the tool path for all its segments, which yields the superior generalisability of this strategy.

To quantitatively measure the shape difference between the target and current workpiece, a curvature difference distribution graph (ΔK -graph) was generated by subtracting the current K_C -graph from the target K_T -graph to represent the workpiece state, as shown in Fig. 3. The current workpiece was considered to be close to its target shape if the value of ΔK approaches zero at any point along the longitudinal length. From the example in Fig. 3, the ΔK -graph was split into 6 segments, A-F. Through the segmental analysis of the ΔK -graphs of real-world components (e.g. aerofoil), three groups of segments were classified, of which any arbitrary ΔK -graph can be composed. Groups 1 and 2 consist of half-wave shaped and quarter-wave shaped segments, and Group 3 includes constant-value segments representing circular arcs or flat sheet.

There are two phases in the generalisable tool path planning strategy, learning phase and inference phase, as shown in Fig. 4. At the learning phase, for each group of segments, m variants of ΔK -graphs, $\Delta K_{i,j}$, were created as shown in Fig. 4a, where i is the group number and j is the variant number. The tool path, $P_{i,j}$, 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

Table 2 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

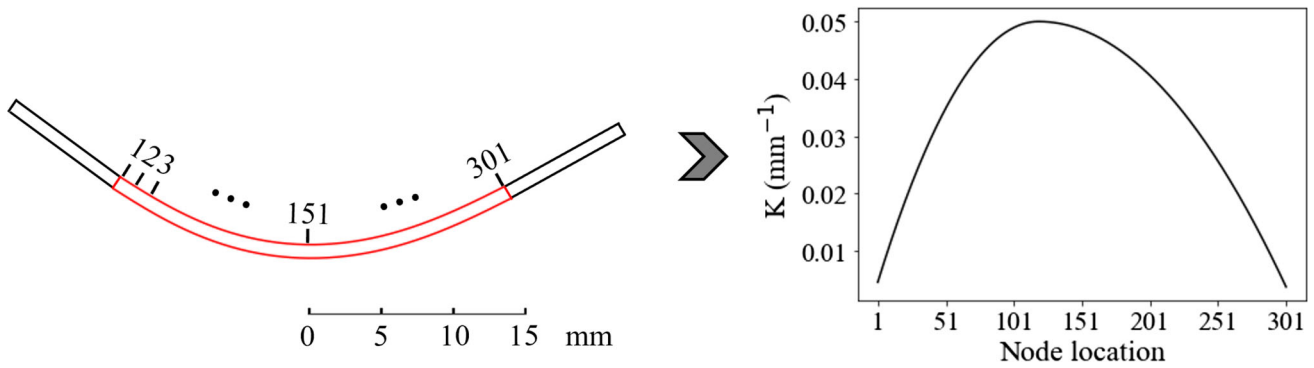


Fig. 2 Example of workpiece shape (left) and its curvature K distribution along node locations (right). The region highlighted in red in the drawing denotes the deformation zone, and the ΔK -graph is generated from the workpiece top surface of this zone

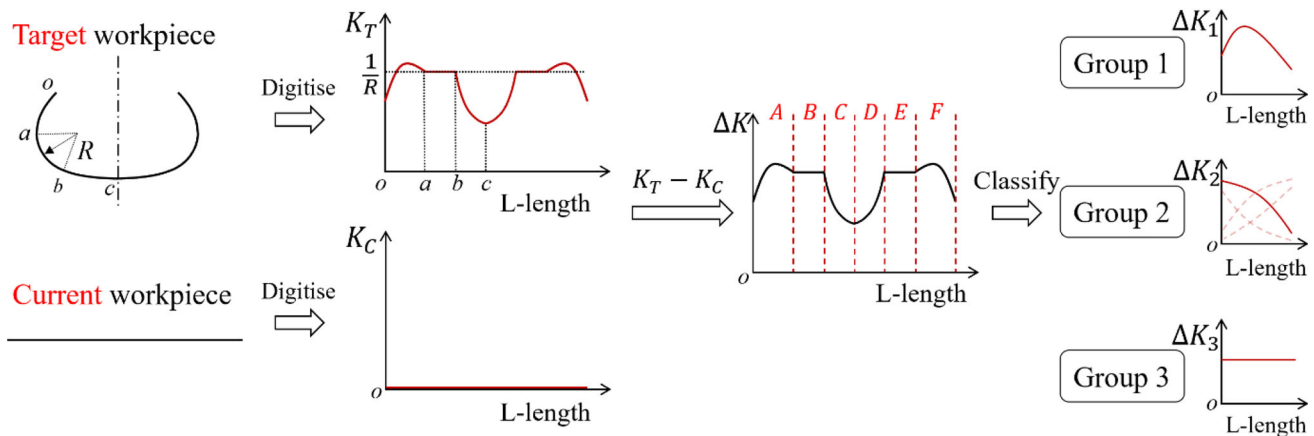


Fig. 3 Schematic digitisation procedure for workpiece state representation ΔK -graph and the classification of three groups of segments. The drawings for target and current workpieces depict their top surfaces.

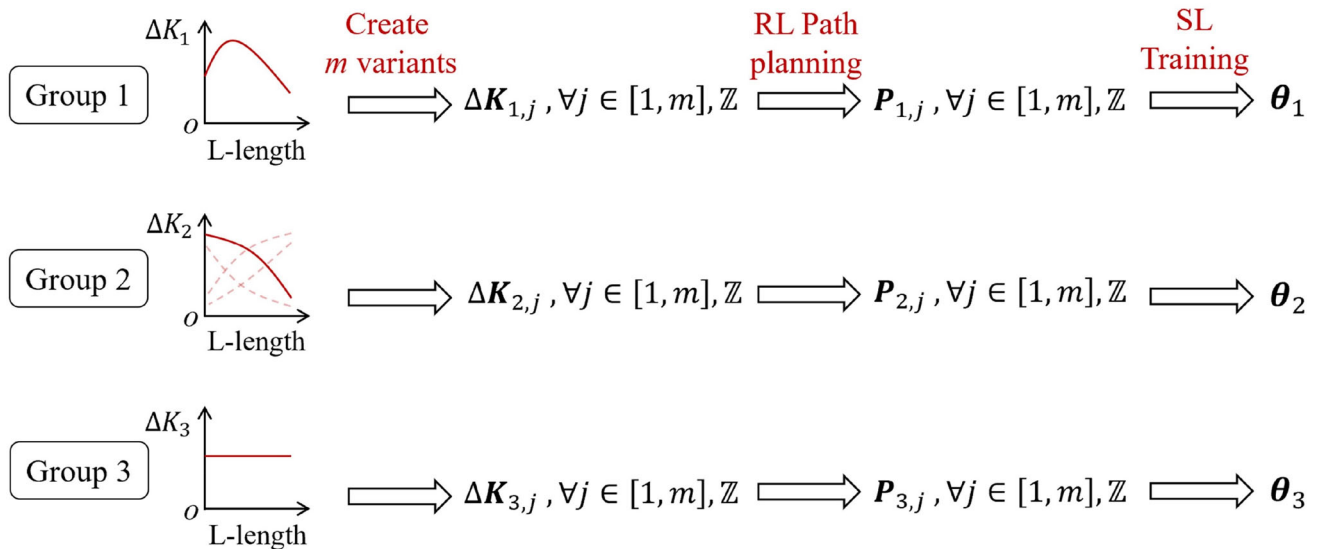
The dashed lines in Group 2 signify other segments having the same shape features as the solid line, which are also counted in this group. L-length denotes longitudinal length

path data, $P_{i,j}$, for each group to generalise the efficient tool path patterns for segments from each group.

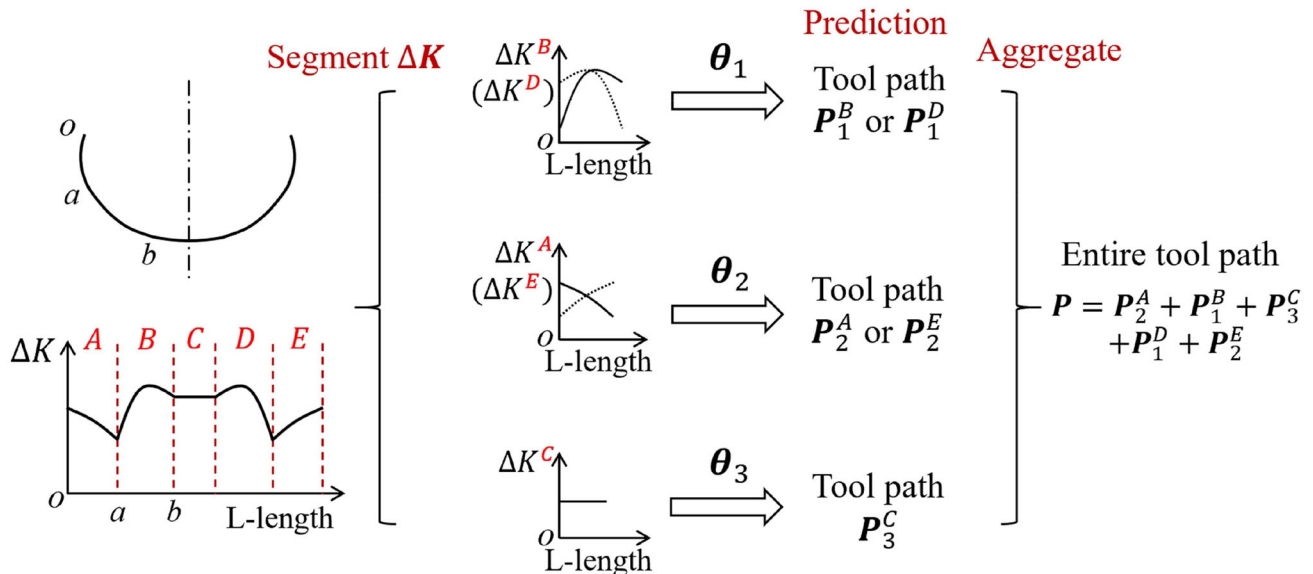
At the inference phase, as shown in Fig. 4b, a new workpiece was firstly digitised to the ΔK -graph and segmented in accordance with the three groups. Five segments, A-E, were obtained in this example, and their tool paths were predicted using the deep supervised learning model trained for their particular 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. To sum up, the RL and SL algorithms were utilised for different purposes in this strategy. The RL model explored the optimal tool path for

each single target workpiece, which was used as the training data of the SL models to learn the efficient forming pattern for a group of workpieces with common features. In application, only SL models were used to infer the tool path of a new workpiece.

In the segmental analysis of the ΔK -graphs, taking the workpiece in Fig. 4b as an example, one can easily find that most segments are from Group 1. The segments from Group 2 can only be seen at the two ends of the components, and the segments from Group 3 only exist in workpiece with circular arc. Thus, Group 1 was used for instantiation of the generalisable tool path planning strategy, and a total of 25



(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) Inference 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.

Fig. 4 The generalisable tool path planning strategy through deep reinforcement and supervised learning

variants of segments in this group were arbitrarily created through the method in Appendix A.

Forming goal and forming parameters design

In the context of the free-form sheet metal stamping test setup presented in Fig. 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 is highest in the ΔK -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 free forming in this research was considered to be achieved if $\max(|\Delta K|) \leq 0.01 \text{ mm}^{-1}$. 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 free forming characteristics. Two phenomena, namely close punch effect 1 and 2 (CPE1 and CPE2), were discovered in this study, which are shown in Fig. 5 and Fig. 6.

From Fig. 5, the ΔK -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 Fig. 5a, b and c. 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.

Figure 6 shows the ΔK -graphs of two workpieces before and after 1 punch and 50 punches, respectively. From Fig. 6a, 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 Fig. 6b, it can be seen that the workpiece had been deformed at location of 118 since the 2nd 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 covers approximately 5 mm (about 50 node locations) around the node location.

From the analysis above, it was found that a stroke of 2.1 mm can reach the forming goal at the 1st punch (with no CPE), and that of at least 3.6 mm was needed to overcome CPE2 and reach the forming goal. Thus, 19 options of punch strokes, ranging from 2.1 to 3.9 mm in 0.1 mm increments, were determined.

Deep reinforcement learning algorithms and learning parameters

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 (Sutton & Barto, 2017):

$$M = \langle S, \mathcal{A}, R, P, \gamma \rangle, \tag{1}$$

where S is a set of possible states, \mathcal{A} is a set of possible actions, R is the reward function, P is the transition probability function and γ is the discounting ratio ($\gamma \in [0,1]$). In this research, S includes the workpiece state representation ΔK and \mathcal{A} includes the options of punch stroke. P is unknown in this research problem, for which model-free RL algorithms are to be applied. The bold capital characters here are used to distinguish them from scalar values in the subsequent equations, such as state or action at a single step.

With the terms introduced above, the RL process can be briefly illustrated with a loop: from the state s_t at time t , an action a_t is based on the current policy, which leads to the next state s_{t+1} and a reward r_t for this step. 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 (Sutton & Barto, 2017):

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

$$Q^\pi(s_t, a_t) = \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t r_t | s_t, a_t, \pi \right], \tag{3}$$

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 (Sutton & Barto, 2017), 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:

$$V^*(s_t) = \max_{a_t \in \mathcal{A}} \mathbb{E}_{s_{t+1}} [r(s_t, a_t) + \gamma V^*(s_{t+1})], \tag{4}$$

$$Q^*(s_t, a_t) = \mathbb{E}_{s_{t+1}} \left[r(s_t, a_t) + \gamma \max_{a_{t+1} \in \mathcal{A}} Q^*(s_{t+1}, a_{t+1}) \right]. \tag{5}$$

Algorithms

Two categories of RL algorithms were investigated in this research, namely value-based and policy-based approaches. When value functions, Eqs. (2) and (3), are approximated with neural network, traditional RL becomes deep reinforcement learning (DRL). For the value-based approaches, three Q-learning algorithms, namely deep Q-learning, double deep Q-learning and dueling deep Q-learning, were implemented. For the policy-based approaches, three policy

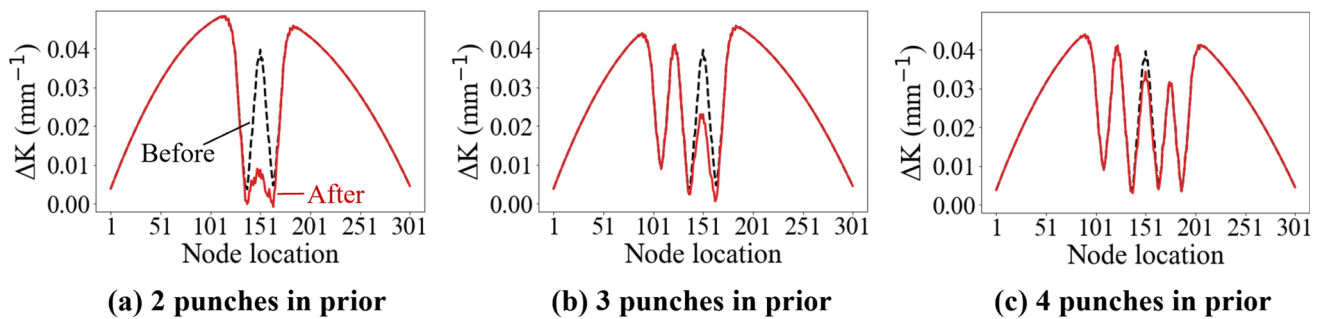


Fig. 5 Close punch effect 1 on the punch with stroke of 3.0 mm at the location of 151. **a**, **b** and **c** present the ΔK -graphs before and after this punch on the workpiece which has been consecutively deformed, in prior, by 2, 3 and 4 punches, respectively

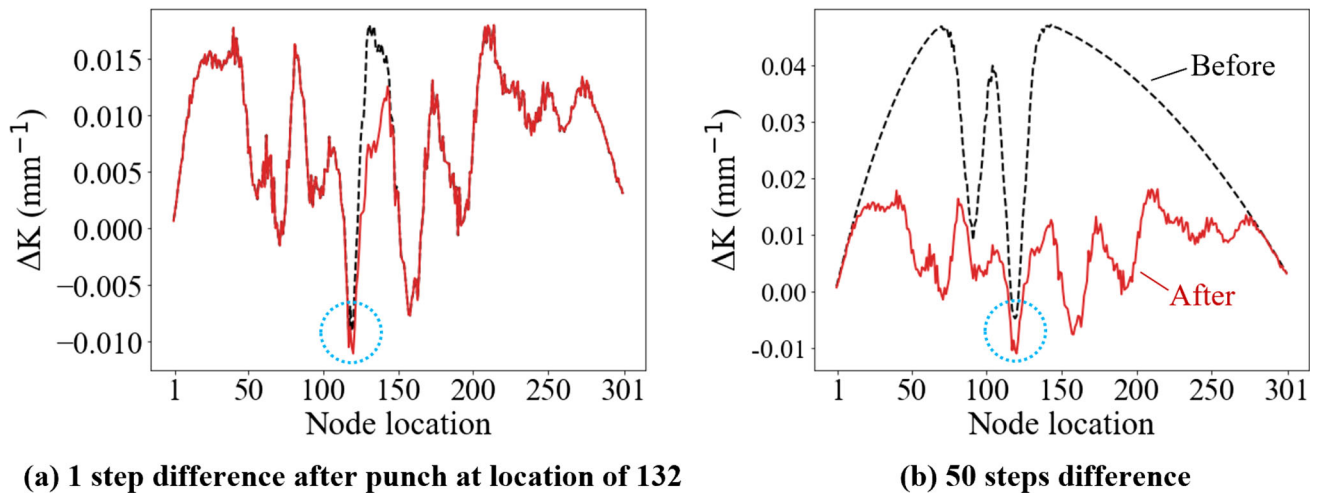


Fig. 6 Close punch effect 2 from the punches near location of 118. **a** and **b** present the ΔK -graphs before and after 1 punch and 50 punches, respectively. The area highlighted by dashed circle is where CPE2 was found

gradient algorithms, namely Advantage Actor-Critic (A2C), Deep Deterministic Policy Gradient (DDPG) and Proximal Policy Optimisation (PPO), were implemented.

As shown in Eq. (5), 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 (Mnih et al., 2015) was brought up to estimate the Q-value function using a function approximator. Three function approximators were investigated in this study: Deep Q-Network (DQN), Double Deep Q-Network (Double-DQN) and Dueling Deep Q-Network (Dueling-DQN), whose objective functions can be found in existing works (Mnih et al., 2015; van Hasselt et al., 2016; Wang et al., 2015). It is noted that Double-DQN alleviates the Q-value overestimation problem for DQN by decomposing the max operation in the target Q-value into two operations of action selection and action evaluation. The Dueling-DQN specifically models the advantage-value, which measures the goodness of an action at a certain state

and is arithmetically related to state-value and action-value by $Q(s, a) = V(s) + A(s, a)$.

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 Eqs. (2) and (3), 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 in (Sutton & Barto, 2017).

The three policy gradient algorithms investigated in this research, A2C, DDPG and PPO, all use an Actor-Critic method (Sutton & Barto, 2017) 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 objective functions can be found in existing works (Mnih et al., 2016; Lillicrap et al., 2015; Schulman

et al., 2017), of which DDPG was specially developed for problems with continuous action space. The A2C algorithm used an advantage term to assist the policy update, while DDPG using the gradient of Q-value with respect to the action and PPO uses Generalised Advantage Estimate (GAE) (Schulman, Moritz, et al., 2015). For A2C, the temporal difference was selected for the advantage estimate through a preliminary study compared with the Monte Carlo (MC) method. The PPO algorithm is a simplified version of Trust Region Policy Optimisation (TRPO) (Schulman et al., 2015a, 2015b) by using a clipped objective function to prevent from extremely large online policy updates and learning instability. The hyperparameters in the PPO algorithm, future advantage discounting ratio and clip ratio, were set as 0.95 and 0.2 in this study following the original work (Schulman et al., 2017).

Learning setup and hyperparameters

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 $\Delta\mathbf{K}$ -graph and a one-hot vector of size 1×301 indicating the punch location. Thus, it represents the shape difference between the current workpiece shape and its target shape, which can be easily used to construct the reward function. The one-hot vector was generated following Heuristic 1.
- The action was the punch stroke, ranging from 2.1 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 Fig. 7. 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 $\Delta\mathbf{K}$ 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: $r_t = 2(p_t/p_o)^2$). An exponential function was used to discourage non-effective punch, since the reward hardly changed at a low punch effectiveness ratio. If the workpiece was overpunched, the $\Delta\mathbf{K}$ at the punch location below the lower threshold -0.01mm^{-1} , $r_t = -100$ (PPO: $r_t = -1$ and DDPG: $r_t = -3$); if the forming goal was achieved, i.e. $\max(|\Delta\mathbf{K}|) \leq 0.01\text{mm}^{-1}$, $r_t = 0$ (PPO: $r_t = 500 - 2.5 \times \text{episode step}$). 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 (Matheron et al., 2019).

Figure 8 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 ended by re-inputting the next state to the agent as the state in the next loop.

The learning methods for the six RL algorithms, which are all model-free algorithms, are shown in Table 3. 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 (Williams & Peng, 1991) 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.

The learning hyperparameters for RL are summarised in Table 4. The maximum step per episode signifies the maximum forming step allowed for each run of the free 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 is softly updated with $\tau = 0.01$ and that for PPO is updated every *rollout* (512 steps in this research) of the online policy. For ϵ -greedy policy, the value of ϵ decays from 1.0 to 0.1.

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 (Lillicrap et al., 2015). Rectified linear unit (*ReLU*) was

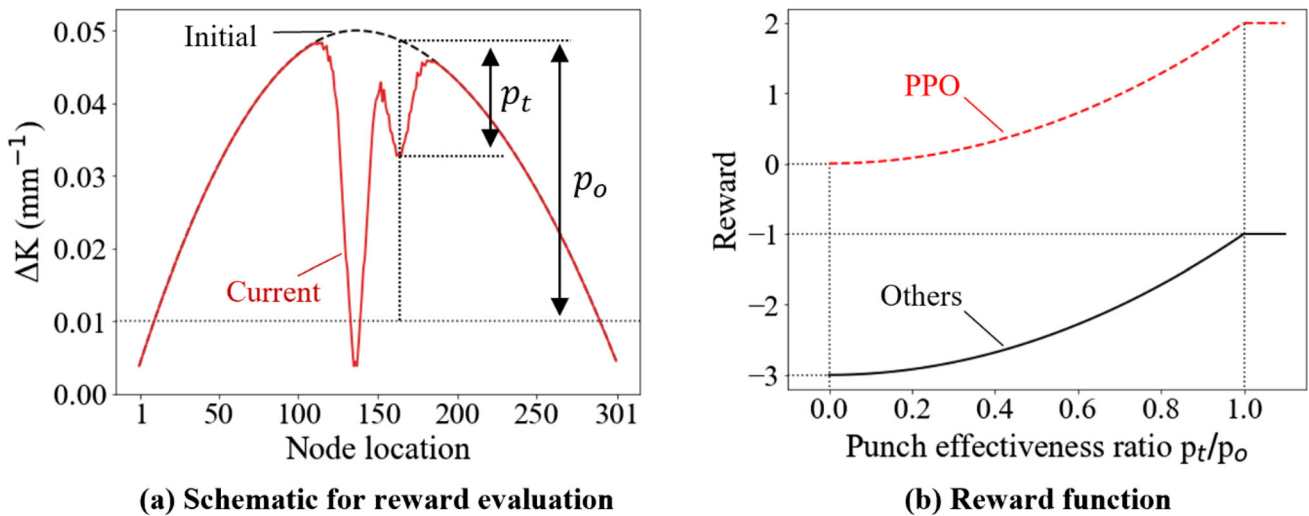


Fig. 7 Reward function and its evaluation for each action. **a** shows the reward evaluation method for punch stroke of 2.2 mm at location of 162, and the two lines denote the initial and current ΔK -graph; **b** shows the reward function for evaluation

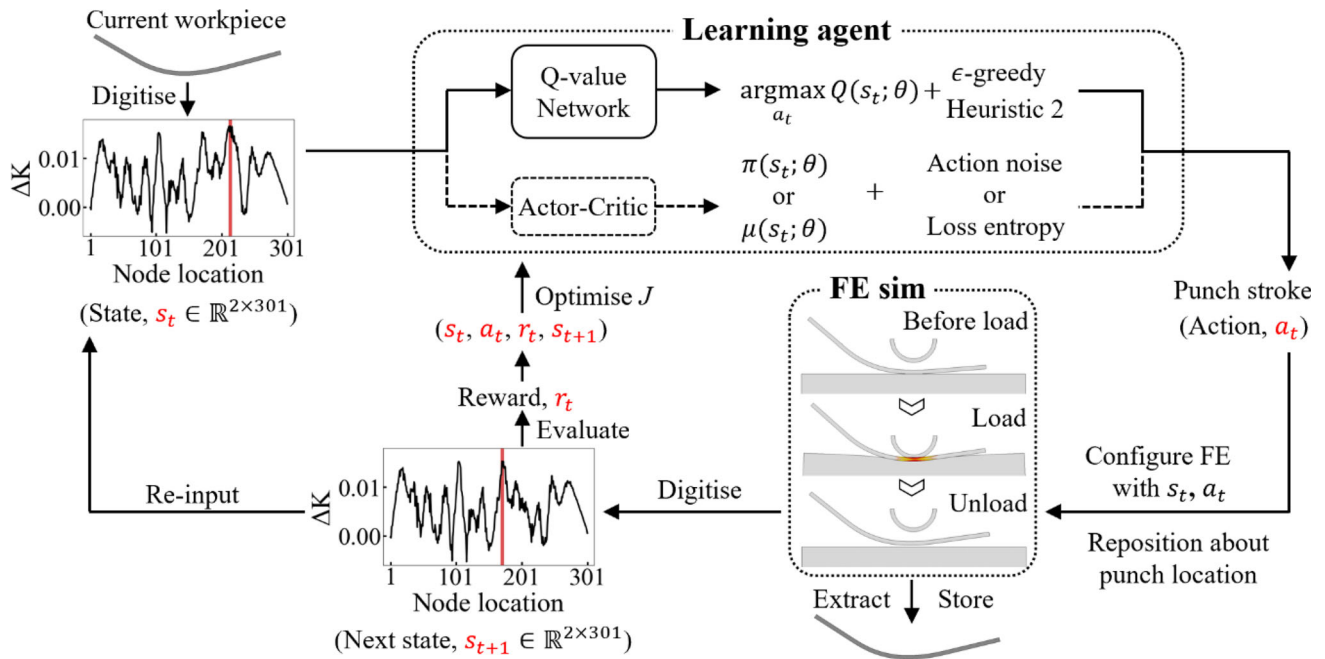


Fig. 8 The reinforcement learning process for the tool path learning of the rubber-tool forming process, of which the FE simulation (FE sim) was used as the RL environment to provide real-time deformation results. The vertical line in the ΔK -graph denotes the punch location

Table 3 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

Table 4 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

used for all hidden layers. There was no activation function for the output layer of the value network, while *softmax* and *tanh* were used for that of the policy network, respectively. The MLP had 2 hidden layers with 400 and 200 units, respectively (164,819 parameters). The network had 2 inputs, the ΔK -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 (Mnih et al., 2015), with an additional hidden layer with 512 units, for the 2nd input, parallel to the last layer of the convolutional layers (1,299,891 parameters).

Virtual environment for RL algorithms comparison

Subject to the FE computational speed, it is considerably time-consuming to test the feasibility of tool path learning for the free forming process using RL. Thus, a virtual environment was developed to imitate the rubber-tool forming behaviour by having similar punch effects on the ΔK -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 "Forming goal and forming parameters design" section, and the effect of stroke value on the ΔK -graph was also imitated by the virtual environment through a parametric study. The detailed setting of the virtual environment is presented in Appendix B.

Deep supervised learning models and training methods

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.

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 the Authors' previous research (Liu et al., 2022). 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 (Simonyan & Zisserman, 2015), ResNet34 and ResNet50 (He et al., 2016) as the feature extractor, respectively. The model architectures for these models were the same as those used in (Liu et al., 2022), 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 ΔK -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 is already contained in the ΔK -graph, it was not fed into the model as a 2nd input, different from (Liu et al., 2022).

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) (Goodfellow et al., 2016) was the objective function for DNN training. Adam algorithm (Kingma & Ba, 2015), with default values of hyperparameters (β_1 , β_2 , ε) 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 (Liu et al., 2022). The key training parameters are shown in Table 5, in which two amounts of training data are presented.

Learning results and discussions

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 rubber-tool forming process. Subjected to the prohibitively expensive FE computation, a virtual environment was developed to imitate the rubber-tool forming behaviour

Table 5 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	$[10^{-6}, 10^{-2}]$

as introduced in "Virtual environment for RL algorithms comparison" section. A total of six RL algorithms were investigated with the data generated by the virtual environment, half of which belong 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 Fig. 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 2.4.2. An additional exploration rule, namely Heuristic 2, was implemented along with the ϵ -greedy policy for Q-learning algorithms. Figure 9 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.

To comprehensively evaluate and compare the performance of the six RL algorithms in tool path learning, four performance factors 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:

$$Avg.Term.Freq. = T_{total} / (S_{final} - S_{first}), \quad (9)$$

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. For this research, the first termination needs to be reached as soon as possible due to the high computational expense. Thus, the importance of 1st Term. Step, Cvg. speed and Avg. Cvg. Term. step is regarded as the same and is greater than that of Avg. Term. Freq.

The six algorithms were trained at four different learning rates using two action/policy function approximators, respectively, as described in Section 2.4.2 on the RL learning setup. The learning performance of each algorithm quantified by the four performance factors is summarised in Tables 6 and 7. 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.

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

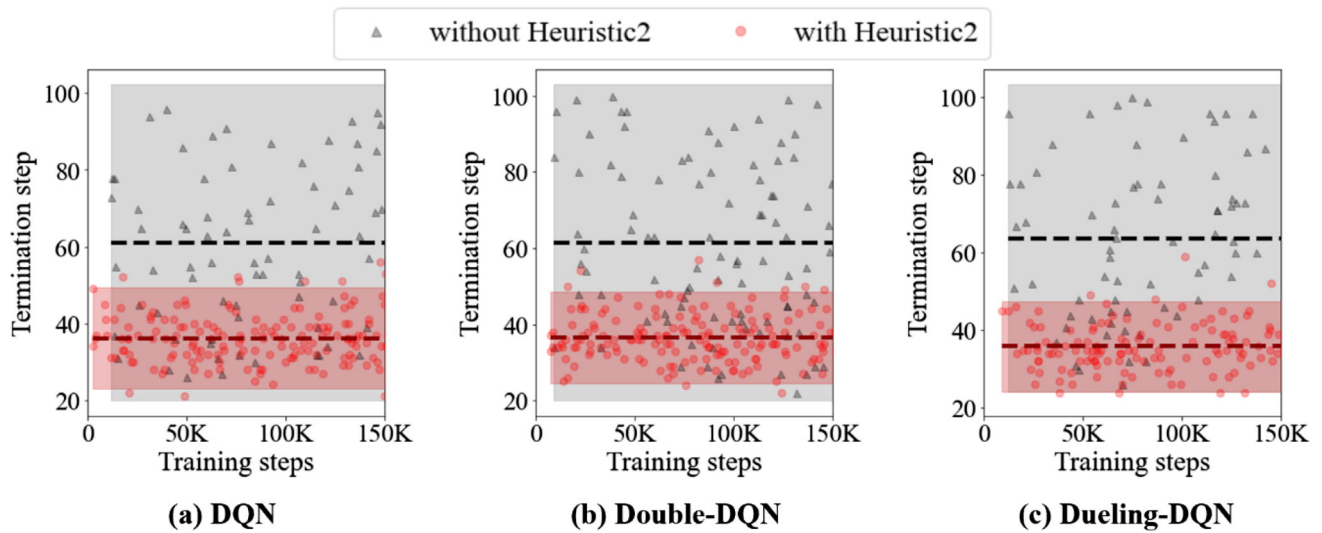


Fig. 9 Comparison of the performances of three Q-learning algorithms trained at 10^{-2} learning rate, with and without heuristic, for tool path learning in terms of termination step. The upper and lower dashed lines denote the average termination step (total punch steps spent to achieve

the forming goal), 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

Table 6 Performance comparison of the Q-learning algorithms for tool path learning

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
Double-DQN	MLP	10^{-4}	52	~ 80	35	2.60
		10^{-1}	78	~ 43	78	–
		10^{-2}	32	~ 3	34	1.28
		10^{-3}	35	~ 80	38	2.23
	CNN	10^{-4}	71	~ 140	42	0.31
		10^{-1}	37	~ 7	33	3.08
		10^{-2}	33	~ 8	37	1.10
Dueling-DQN	CNN	10^{-3}	49	~ 80	36	2.77
		10^{-4}	50	~ 140	33	3.39
		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

Term. denotes termination. *Cvg.* denotes converge. *Freq.* denotes frequency. *Avg.* denotes average “–” denotes less than 3 terminations occur in the training

Numbers in bold font denote the best tweaking results for each algorithm

Table 7 Performance comparison of the policy gradient algorithms for tool path learning

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
		10^{-2}	49	~ 60	29	23.21
DDPG	MLP	10^{-1}	80	~ 133	80	–
PPO	MLP	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
	10^{-2}		51	~ 200	35	28.25
	10^{-3}		75	~ 3500	37	0.67
	10^{-4}		54	~ 2300	37	9.39

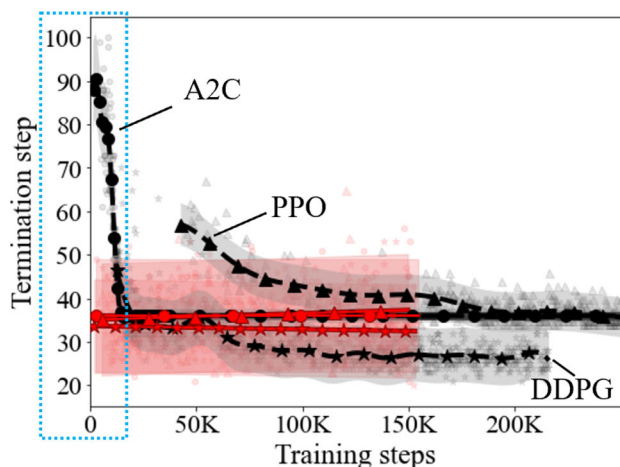
Term. denotes termination. *Cvg.* denotes converge. *Freq.* denotes frequency. *Avg.* denotes average “–” denotes less than 3 terminations occur in the training
Numbers in bold font denote the best tweaking results for each algorithm

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.

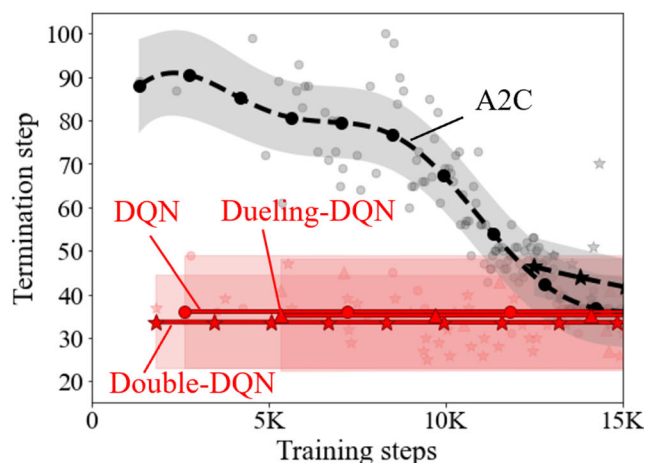
Figure 10 shows the training process of the six RL algorithms, which were trained with the best hyperparameters from above. It can be seen that, unlike the Q-learning algorithms which almost instantly converged after a few terminations, the policy gradient ones had more discernible converging process. 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. The Q-learning algorithms outperformed the policy gradient ones in general in terms of the first termination step and convergence speed. This could be due to that A2C and PPO are on-policy learning which is less data-efficient than off-policy, and DDPG is created for learning problems with continuous action space which needs careful tuning for problems with discrete space. For Q-learning algorithms, the Double-DQN preceded DQN and Dueling-DQN for its lower average converge termination step and marginally faster first termination. The reason could be that Double-DQN alleviates the Q-value over-estimation problem in DQN learning, and Dueling-DQN is only particularly useful when the relevance of actions to the goal can be differentiated by separately learning state-value and advantage-value. However, each action in free-form deformation is highly relevant to the goal, for which the structure of Dueling-DQN, in turn, increases learning complexity and slows down learning speed. Thus, for the following results, Double-DQN was used to learn the optimal tool path.

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 Fig. 11. From Fig. 11a and b, the history of forming step and total rewards per episode from the learning using VE have the



(a) Whole learning process



(b) Detailed view in dotted square

Fig. 10 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

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 is more complex and nonlinear than the VE imitates. From Fig. 11c, both tool paths had similar forming pattern of alternately selecting small and large stroke values which have, on average, slight 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.

Tool path learning results for 25 workpieces using double-DQN

From Section 3.1, 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 Fig. 12. The workpieces were deformed through the rubber-tool forming process, which was simulated through FE computations. The K -graphs were arbitrarily created with the method shown in Appendix A, and the real-scale shapes

from Gaussian process regression, through the training process of the algorithms. The shaded regions denote 95% confidence interval

in Fig. 12b were reconstructed from the K -graphs using constant initial interval between two contiguous node locations (0.1 mm).

An exemplary Double-DQN learning process is shown in Fig. 11a and b, where the termination occurs 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 ΔK value at the end of each episode is shown in Fig. 13. As the forming goal is to achieve a workpiece state where its $\max(|\Delta K|) \leq 0.01 \text{ mm}^{-1}$, the learning history of episode end maximum ΔK can reflect the learning progress of effective tool path. From Fig. 13, there is a clear trend that the maximum value of ΔK 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. This learning curve demonstrates both the effectiveness of the Double-DQN algorithm and the reward function in searching the tool path. With progressing, the deformed sheet metal was more and more approaching its target shape, which was demonstrated by the troughs of the $\max(\Delta K)$ graph along the arrow marker.

In addition to the learning progress captured from maximum ΔK 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 14

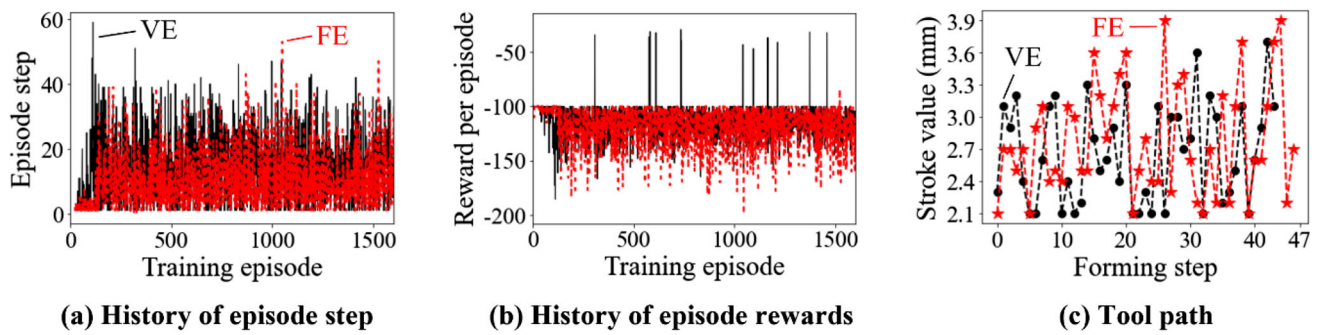


Fig. 11 Comparison between RL using virtual environment (VE) and FE simulation environment (FE) in terms of **a** the history of steps in each episode, **b** the history of total rewards in each episode and **c** the tool path predictions

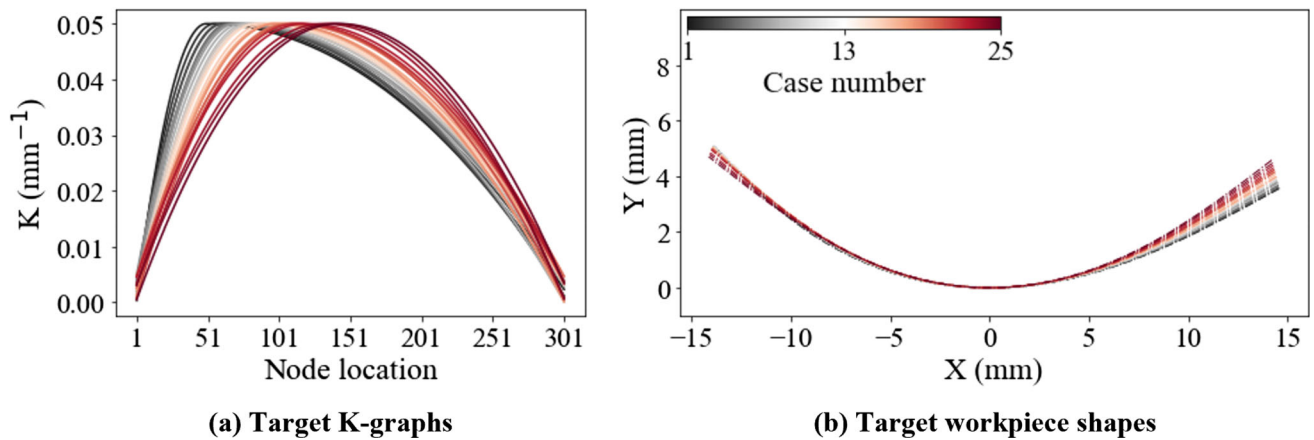


Fig. 12 The K -graphs and workpiece shapes for all the generated target workpieces

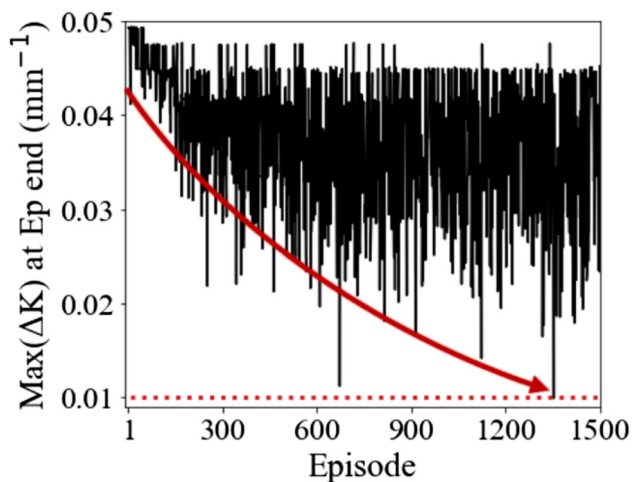


Fig. 13 The history of the maximum ΔK value at the end of each episode (Ep) throughout the learning process. The arrow shows the learning progress of effective tool path

shows two examples where the self-learning characteristics of tool path efficiency improvement and overpunch circumvention were captured, respectively. The three ΔK -graphs were collected from the workpiece deformed by the same

number of punches at different episodes during a learning process. In Fig. 14a, 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 ΔK -graph area. In Fig. 14b, 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.

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,

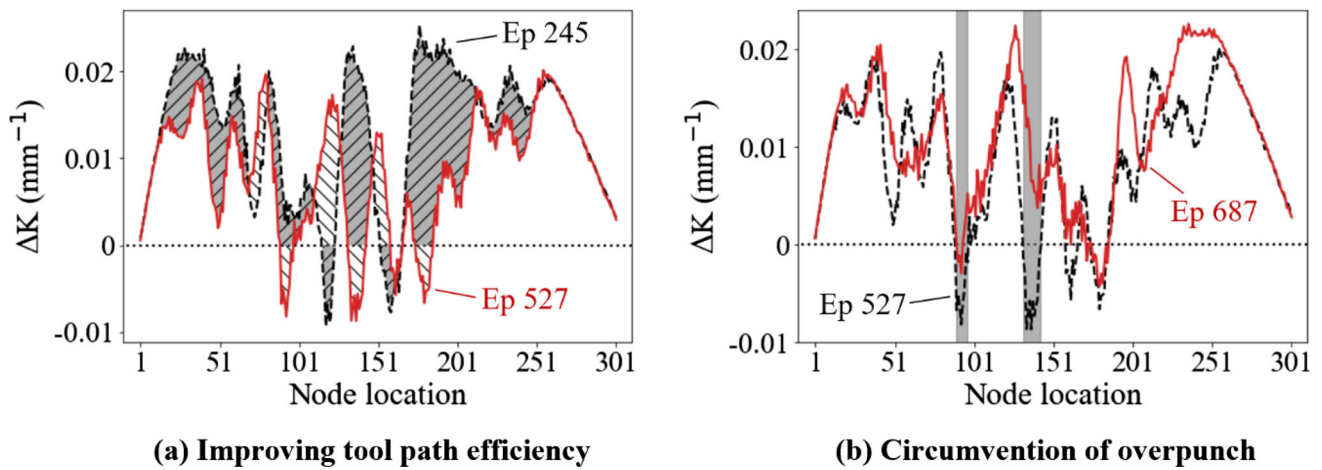


Fig. 14 Examples showing self-learning characteristics of **a** improving tool path efficiency and **b** circumvention of overpunch. The results were from step 31 at three different episodes of the tool path learning process for a workpiece

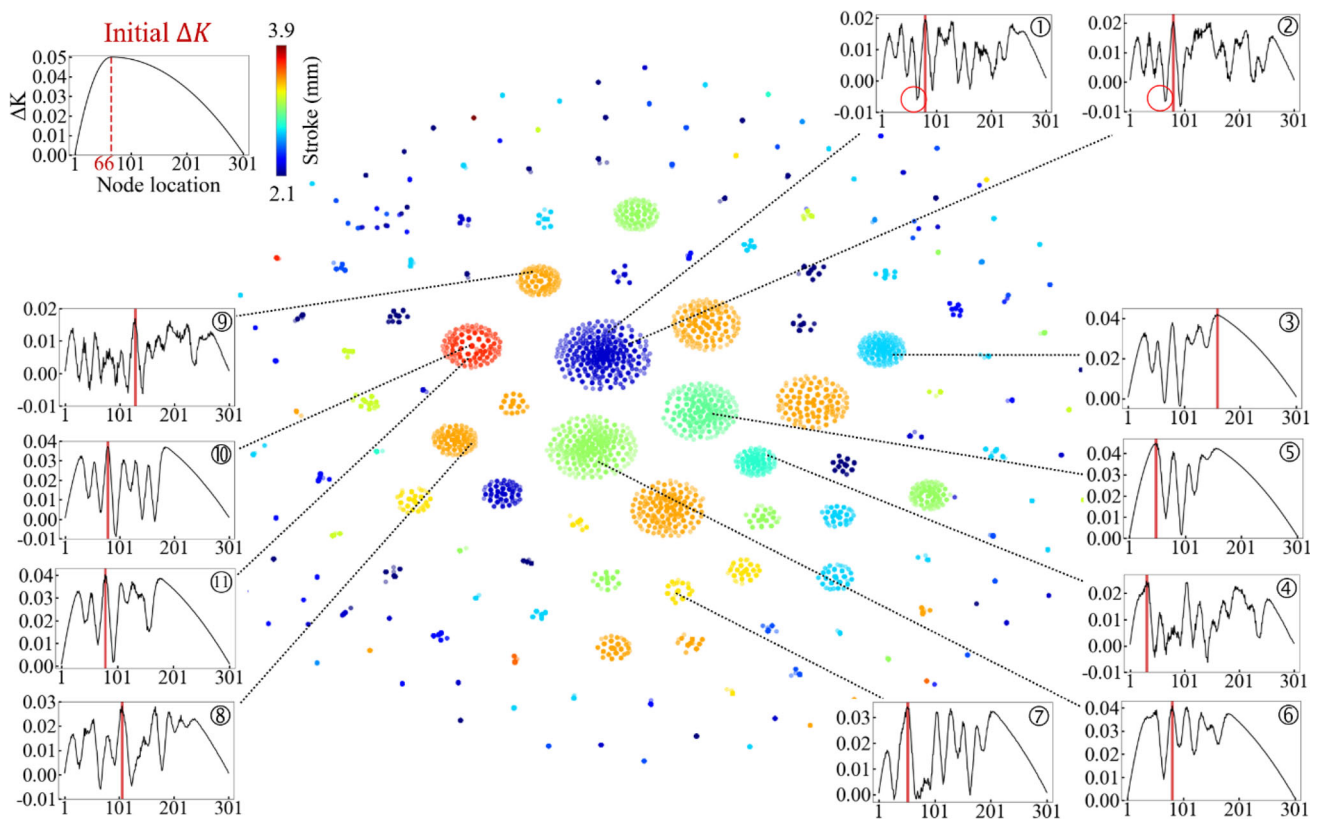


Fig. 15 The two-dimensional embedding, generated through t-SNE, of the representations in the last hidden layer of the Double-DQN to workpiece states (ΔK -graphs) experienced during tool path learning. The points are coloured according to stroke values selected by the agent.

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

were retrieved and reduced to two-dimensional embeddings through t-SNE technique (Maaten & Hinton, 2008). The visualisation of these embeddings is shown in Fig. 15, 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 Fig. 15, the ΔK -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 ΔK -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 ΔK -graphs ① and ②, where the effect was significantly more obvious than the one shown in Fig. 6a. 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.

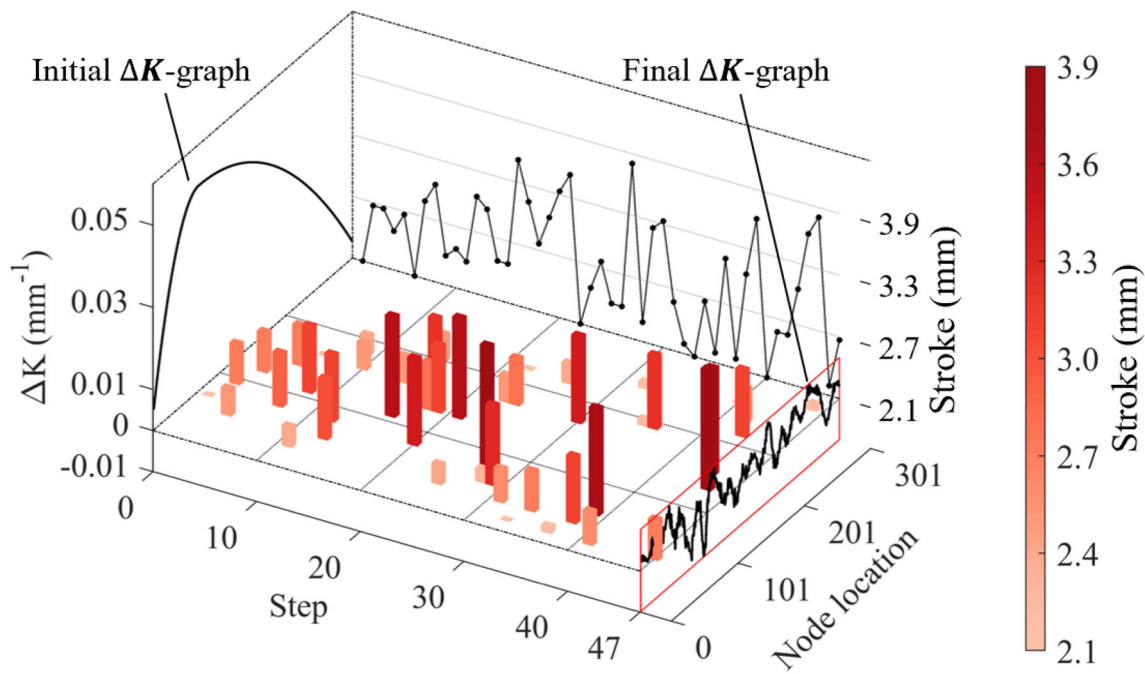
Figure 16 shows an example of the tool path learned by the Double-DQN. In Fig. 16a, the initial ΔK -graph between the blank sheet and the target workpiece was transformed to the final one (enclosed by a rectangle), where the ΔK values at all node locations were within the forming thresholds, by 47 forming 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 Fig. 16b. 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 Fig. 16a. 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 Fig. 16b, large strokes were concentratedly assigned to punch locations with high initial ΔK values and descended to those with low ones.

Figure 17 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.

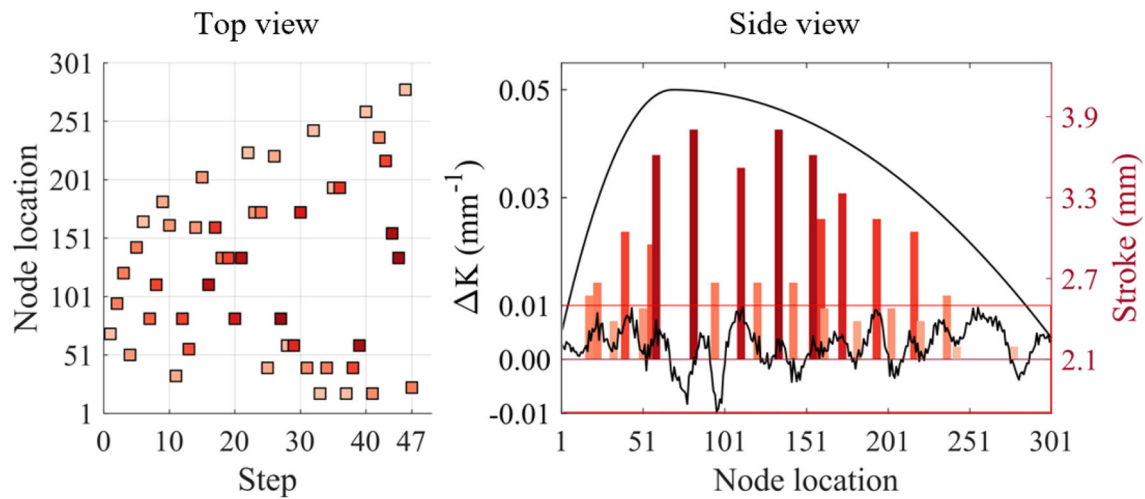
Tool path learning generalisation using supervised learning

Through Double-DQN algorithm, the tool paths for the 25 variants of workpiece (shown in Fig. 12) segment were learned, whose length (total punch steps) for each variant is shown in Fig. 18. The tool path lengths varied from 44 to 63, with most lying around 52.

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 "Deep neural networks" section, 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 split into 90% for training and 10% for testing, and the other key training parameters are presented in Table 5. The training processes of the three models are shown in Fig. 19 by generalisation loss (test loss) history, which would end early if the generalisation loss tended to increase (Goodfellow et al., 2016). 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.



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



(b) Different views

Fig. 16 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

It is also noted that the loss from LSTMs with both ResNets sharply decreased before convergence, which could be due to the decaying learning rate during training. Before learning rate decreased to a certain level, the parameter update at each learning step could be so large that parameter value was jiggling around its suboptimum. Once the learning rate became smaller than this level, the model parameters could be closer

to their optimal values and the loss would encounter a sharp drop.

Figure 20 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. The total time steps of the LSTMs were the maximum forming steps in the training data, namely 63 and 55 steps for models trained with 25 and 20 tool paths, respectively. It can be seen that the

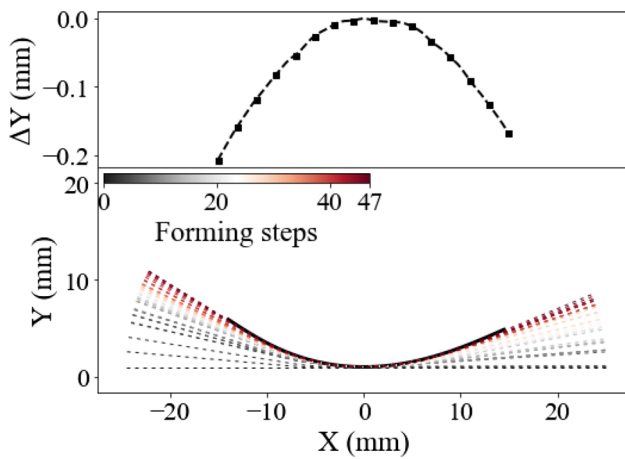


Fig. 17 An example of workpiece deformed by the tool path predicted by the Double-DQN. Top: the dimension error between the target workpiece and the final workpiece deformed along the predicted tool path. Bottom: the workpiece shape at each forming step (dotted line) compared to its target shape (solid line). Forming step of 0 denotes blank sheet

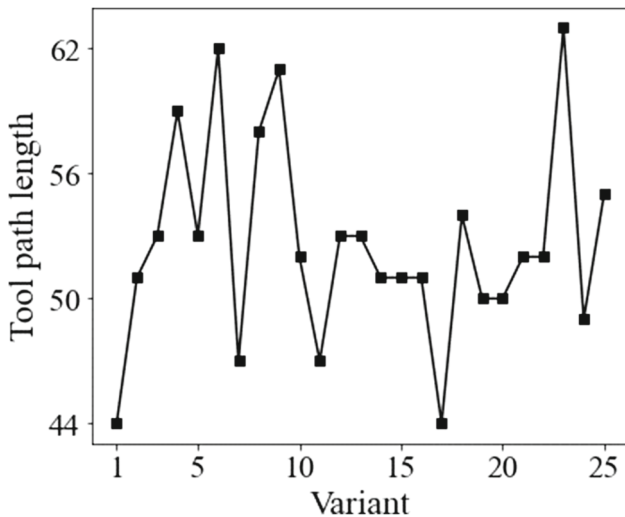


Fig. 18 The length of the tool path learned through Double-DQN for each variant of workpiece segment

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 alternately 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 ΔK -graph of the test workpiece deformed through the tool path predicted by the LSTMs,

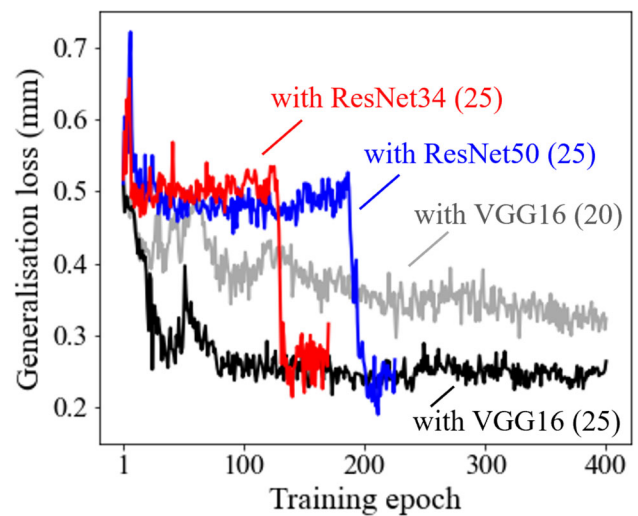


Fig. 19 The generalisation loss curves along the training processes of the LSTM models with VGG16, ResNet34 and ResNet50 as the feature extractor, respectively. The number 25 and 20 in the parenthesis denote the amount of tool paths used for training

the VGG16 LSTM trained with 25 tool paths was the most superior among all models, whose level of forming goal achievement ($G = 1 - \Delta K_{final}^{out THLD} / \Delta K_{initial}^{out THLD}$, 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 ΔK -graphs from models trained with 25 tool paths in Fig. 20, 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 ΔK -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 ΔK -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

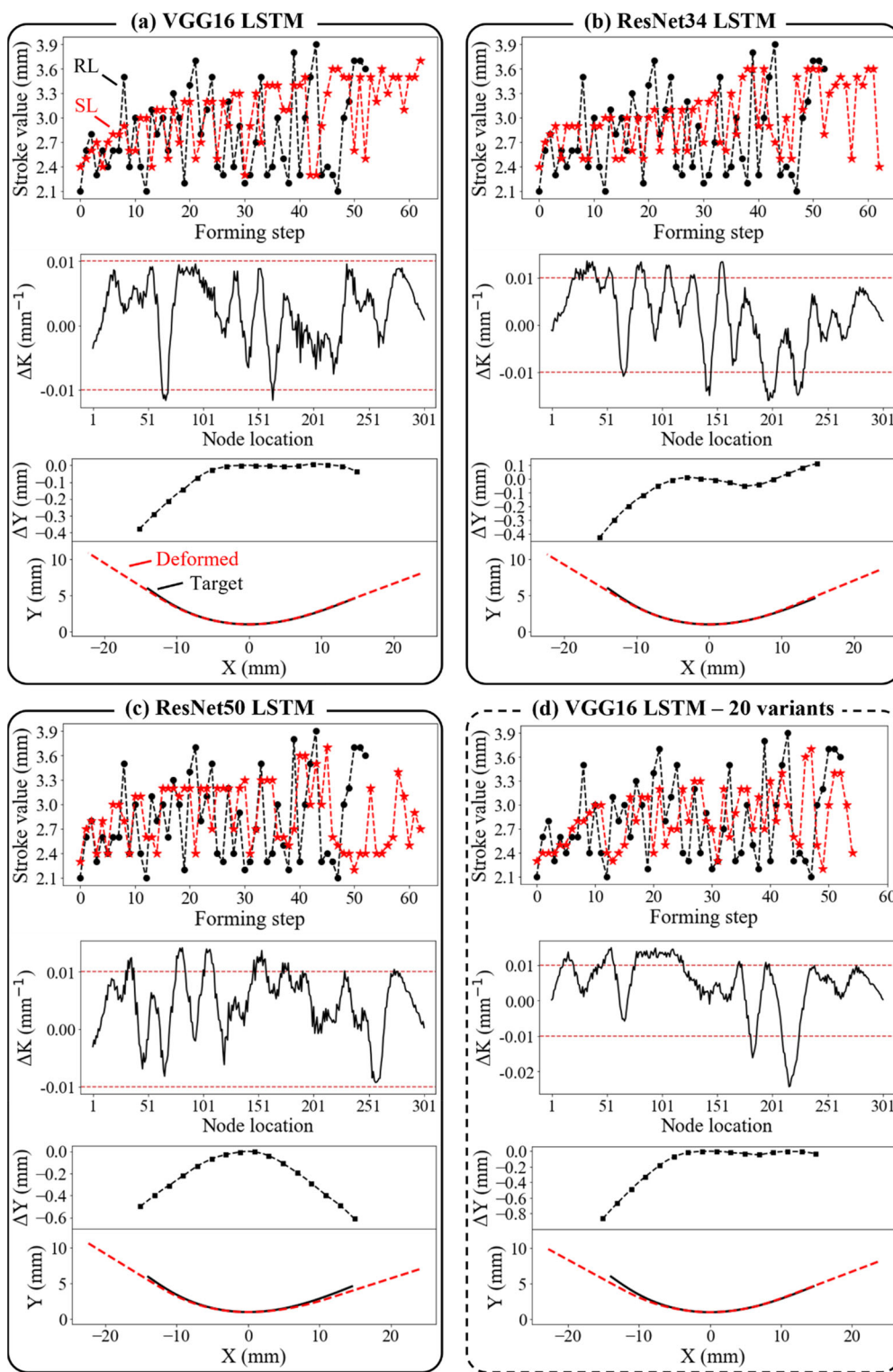


Fig. 20 The learning performance of LSTMs trained with 25 tool paths data (solid squares) and 20 tool paths data (dashed squares) on a test workpiece. For the former, the prediction results from LSTMs with **a** VGG16, **b** ResNet34 and **c** ResNet50 are presented, while for the latter, that from **d** 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

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 ΔK -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 ΔK -graph had a good goal achievement, which is due to the excessively more area of ΔK -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 ΔK 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 ΔK -graphs (i.e., different target shapes), including the target shape it was trained for.

Figure 21 shows the ΔK -graphs after the first punch of these workpieces with the stroke prediction from the Double-DQN, of which the node locations where the troughs reside

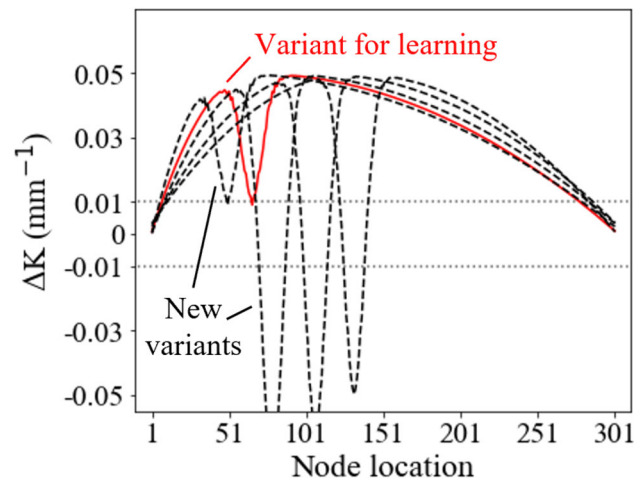


Fig. 21 Evaluation of the pure RL strategy by assessing the tool path prediction results from the trained Double-DQN model for new variants of workpiece. The ΔK -graphs were acquired after the first punch, predicted by the RL model, for the variant used for tool path learning through RL (solid line) and new variants that were never seen in the learning process (dashed line)

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.

Case study verifying the generalisable tool path planning strategy

To evaluate the generalisable tool path planning strategy presented in Fig. 4, a new target workpiece of length 90.2 mm was arbitrarily generated as shown in Fig. 22. The target workpiece was first digitised to its initial ΔK -graph using 0.1 mm interval between two contiguous node locations, with 903 node locations in total. The ΔK -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 ΔK -graph resided well in the threshold region with the level of forming goal achievement of 99.87%. Thus, the case study verifies the generalisation of the proposed strategy that an arbitrarily selected workpiece can be formed by solving its tool path in a dynamic programming way. By factorising

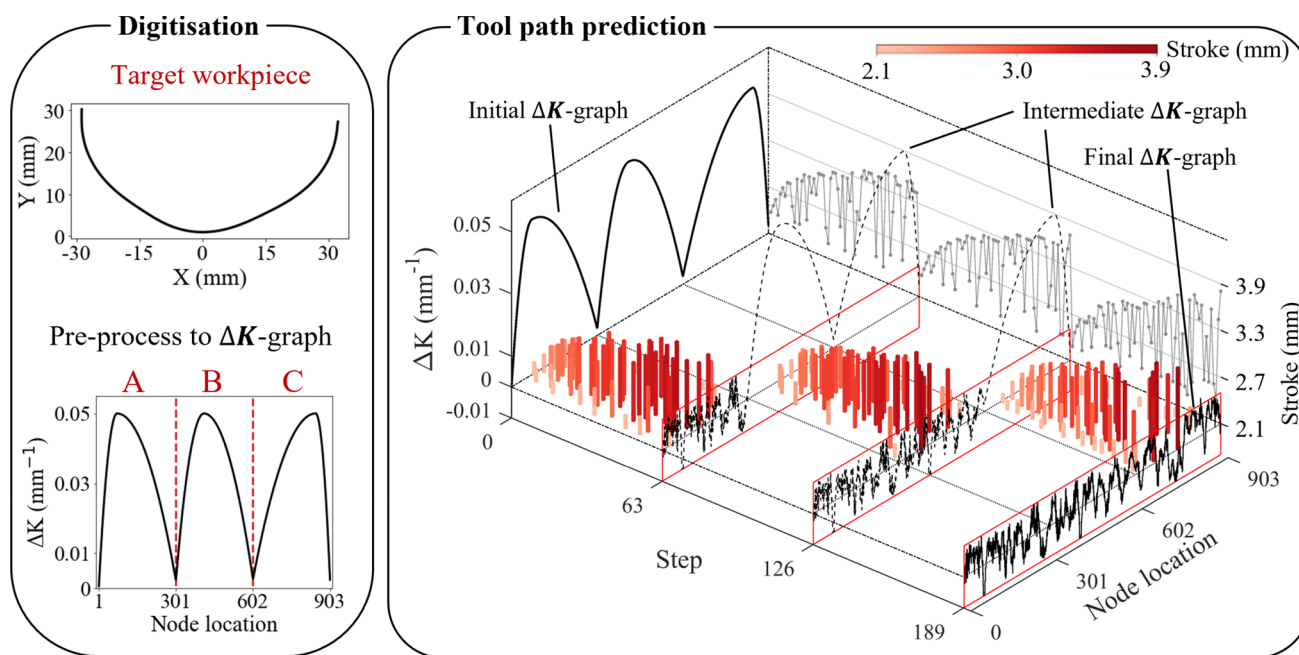


Fig. 22 The entire tool path for a new workpiece, which is aggregated by the subpaths predicted by the VGG16 LSTM for each segment of the workpiece. The three segments were never seen in the LSTM training

the forming process of an entire workpiece into that of typical types of segments, the entire workpiece can be formed by consecutively forming each segment.

Figure 23 presents the workpiece shape after deformation, computed by FE, through the generalisable tool path planning strategy and its target shape. From Fig. 23a, due to the accumulation of ΔK -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 is 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.

Conclusions

In this research, a generalisable tool path planning strategy for free-form sheet metal stamping was proposed through deep reinforcement and supervised learning technologies.

By factorising the forming process of an entire workpiece into that of typical types of segments, the tool path planning problem was solved in a dynamic programming way, which yielded a generalisable tool path planning strategy for a curved component for the first time. RL algorithms and SL models were exploited in tool path learning and generalisation, and six deep RL algorithms and three deep SL models were investigated for performance comparison. The proposed strategy was verified through an application to a case study where the forming tool path for a completely different target workpiece from training data was predicted. From this study, it can be concluded that:

- (1) Q-learning algorithms are superior to policy gradient algorithms in tool path planning of free-form sheet metal stamping process, in which Double-DQN precedes DQN and Dueling-DQN. The forming heuristic is also corroborated to further improve the Q-learning performance.
- (2) 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 free-form sheet metal stamping process can be successfully acquired, with the dimension error of the deformed workpiece below 0.26 mm (0.87%).

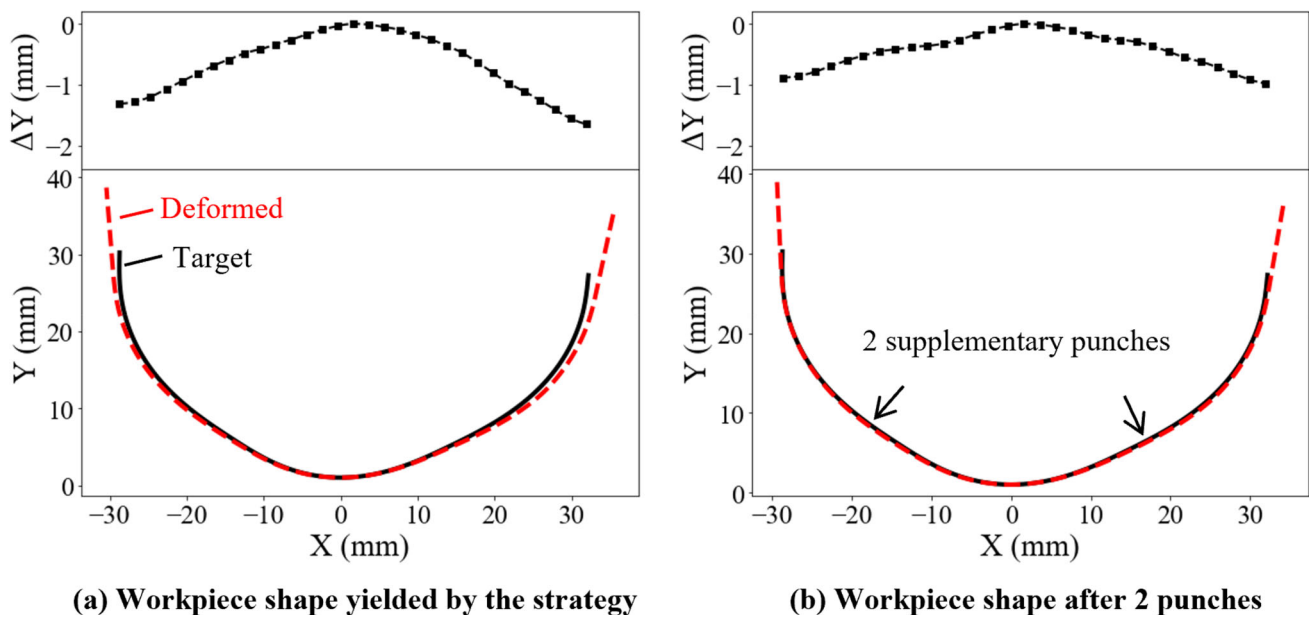


Fig. 23 The comparison between the workpiece shape deformed through the generalisable tool path planning strategy and its target shape. **a** deformed workpiece shape yielded by the strategy and **b** workpiece

shape after two supplementary punches near the junctions of the three workpiece segments

- (3) 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.
- (4) 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 reached 99.87% and the dimension error of the deformed workpiece was 2%. The dimension error could be reduced to about 1.1% with two small supplementary punches near the junctions of the workpiece segments.

Through the proposed method, the tool path planning for an arbitrary sheet metal component is attempted with a generalisable strategy for the first time, and the poor generalisation issue of pure reinforcement learning approach for tool path planning is addressed. However, the efficiency of this strategy is subject to the design of forming the pattern and reward function. In future work, a multi-objective forming goal for tool path planning could be used for 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 might be yielded. CPE1 and CPE2 can also be embedded into the reward function design to facilitate the tool path learning process.

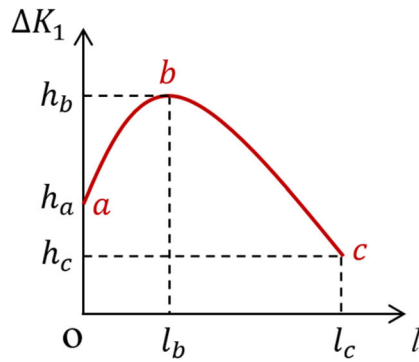
Appendix A: arbitrary generation of workpiece segments

The ΔK -graph of each variant in Group 1 shown in Fig. 24, which is composed of two parabolas ΔK_{a-b} and ΔK_{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). The workpiece segments were arbitrarily generated by randomly sampling the values of these variables.

Appendix B. virtual environment configuration

The virtual environment (VE) is developed to imitate the rubber-tool forming behaviour 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,

Fig. 24 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



- $h_b: 0.05 \text{ mm}^{-1}$
- $h_a: 0 - 0.005 \text{ mm}^{-1}$
- $h_c: 0 - 0.005 \text{ mm}^{-1}$
- $l_b: 5 - 15 \text{ mm}$
- $l_c: 30 \text{ mm}$
- $\Delta K_{a-b} = w_{ab}(l - l_b)^2 + h_b$
- $\Delta K_{b-c} = w_{bc}(l - l_b)^2 + h_b$

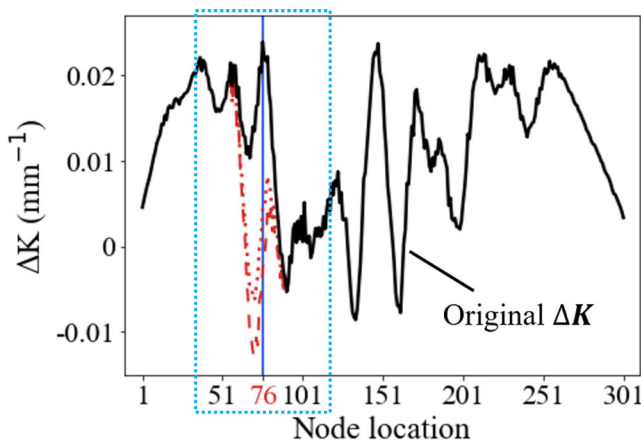
whose formulation and parameter selection were based on FE simulation results.

1. A single punch operation only affects the ΔK values at 50 node locations (5 mm) around the punch location and the punch location itself in the ΔK -graph.
2. If a node location has been punched with a stroke and this location is selected again for punching, the ΔK -graph will only change if the new stroke is greater than the previous one.
3. The change of 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: $c_{K3} = (c_{K0} + 0.005)/2$;

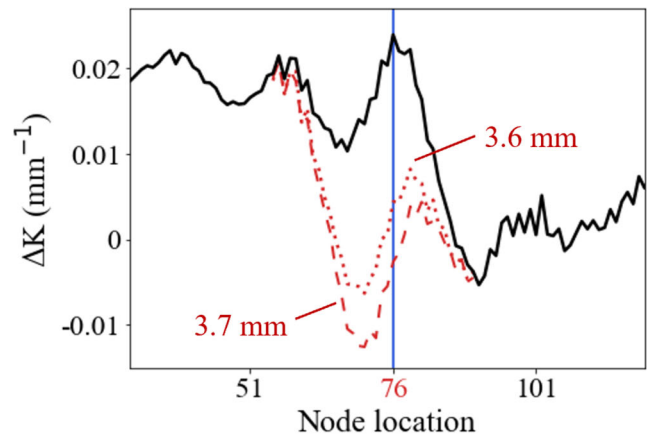
- iv. two sides of the punch locations are pre-deformed by 4 and over 4 punches in total: $c_{K4} = (c_{K0} - 0.01)/2$.
5. The change of K value gradually decreases from c_{Ki} ($i \in [0,4], \mathbb{Z}$) at the punch location to 0 at two ends of the 51 node locations in rule 1.
6. With CPE2: if the ΔK -graph is changed by the punch, ΔK values at the 51 node locations in rule 1 are reduced by 0.0005 mm^{-1} .

Appendix C: extraordinarily large CPE2

A phenomenon of extraordinarily large CPE2 is shown in Fig. 25. 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} .



(a) Original view



(b) Detailed view in dotted square

Fig. 25 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

Acknowledgments S. Liu is grateful for the support from China Scholarship Council (CSC) (Grant no. 201908060236).

Funding S. Liu received subsistence allowance from China Scholarship Council (CSC) under Grant no. 201908060236.

Data availability The data that support the findings of this study are available from the corresponding author upon request.

Declarations

Competing interest The authors have no competing interests to declare that are relevant to the content of this article.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Allwood, J. M., & Utsunomiya, H. (2006). A survey of flexible forming processes in Japan. *International Journal of Machine Tools and Manufacture*, 46(15), 1939–1960. <https://doi.org/10.1016/j.ijmachtools.2006.01.034>.
- Attanasio, A., Ceretti, E., & Giardini, C. (2006). Optimization of tool path in two points incremental forming. *Journal of Materials Processing Technology*, 177(1–3), 409–412. <https://doi.org/10.1016/j.jmatprotec.2006.04.047>.
- Azaouzi, M., & Lebaal, N. (2012). Tool path optimization for single point incremental sheet forming using response surface method. *Simulation Modelling Practice and Theory*, 24, 49–58. <https://doi.org/10.1016/j.simpat.2012.01.008>.
- Bowen, D. T., Russo, I. M., Cleaver, C. J., Allwood, J. M., & Loukaides, E. G. (2022). From art to part: Learning from the traditional smith in developing flexible sheet metal forming processes. *Journal of Materials Processing Technology*, 299, 117337. <https://doi.org/10.1016/j.jmatprotec.2021.117337>.
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. The MIT.
- Hartmann, C., Opritescu, D., & Volk, W. (2019). An artificial neural network approach for tool path generation in incremental sheet metal free-forming. *Journal of Intelligent Manufacturing*, 30(2), 757–770. <https://doi.org/10.1007/s10845-016-1279-x>.
- He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), (pp. 770–778). Las Vegas, NV, USA. <https://doi.org/10.1109/CVPR.2016.90>
- Kingma, D. P., & Ba, J. L. (2015). Adam: a method for stochastic optimization. [arXiv:1412.6980](https://arxiv.org/abs/1412.6980)
- Kirk, R., Zhang, A., Grefenstette, E. and Rocktäschel, T. (2021). A survey of zero-shot generalisation in deep reinforcement learning. [arXiv:2111.09794](https://arxiv.org/abs/2111.09794)
- Kubik, C., Knauer, S. M., & Groche, P. (2022). Smart sheet metal forming: Importance of data acquisition, preprocessing and transformation on the performance of a multiclass support vector machine for predicting wear states during blanking. *Journal of Intelligent Manufacturing*, 33(1), 259–282. <https://doi.org/10.1007/s10845-021-01789-w>.
- Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., & Wierstra, D. (2015). Continuous control with deep reinforcement learning. <https://arxiv.org/abs/1509.02971>
- Liu, S., Shi, Z., Lin, J., & Li, Z. (2020). Reinforcement learning in free-form stamping of sheet-metals. *Procedia Manufacturing*, 50, 444–449. <https://doi.org/10.1016/j.promfg.2020.08.081>.
- Liu, S., Xia, Y., Liu, Y., Shi, Z., Yu, H., Li, Z., & Lin, J. (2022). Tool path planning of consecutive free-form sheet metal stamping with deep learning. *Journal of Materials Processing Technology*, 303, 117530. <https://doi.org/10.1016/j.jmatprotec.2022.117530>.
- Liu, S., Xia, Y., Shi, Z., Yu, H., Li, Z., & Lin, J. (2021). Deep learning in sheet metal bending with a novel theory-guided deep neural network. *IEEE/CAA Journal of Automatica Sinica*, 8(3), 565–581. <https://doi.org/10.1109/JAS.2021.1003871>.
- Low, D. W. W., Chaudhari, A., Kumar, D., & Kumar, A. S. (2022). Convolutional neural networks for prediction of geometrical errors in incremental sheet metal forming. *Journal of Intelligent Manufacturing*. <https://doi.org/10.1007/s10845-022-01932-1>.
- Malhotra, R., Bhattacharya, A., Kumar, A., Reddy, N. V., & Cao, J. (2011). A new methodology for multi-pass single point incremental forming with mixed toolpaths. *CIRP Annals*, 60(1), 323–326. <https://doi.org/10.1016/j.cirp.2011.03.145>.
- Matheron, G., Perrin, N., & Sigaud, O. (2019). The problem with DDPG: understanding failures in deterministic environments with sparse rewards. [arXiv:1911.11679](https://arxiv.org/abs/1911.11679)
- Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D., & Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. *Proceedings of the 33rd International Conference on Machine Learning, PMLR 48*, 1928–1937. <https://proceedings.mlr.press/v48/mnih16.html>
- Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540), 529–533. <https://doi.org/10.1038/nature14236>.
- Monostori, L., Markus, A., Van Brussel, H., & Westkämpfer, E. (1996). Machine learning approaches to manufacturing. *CIRP Annals*, 45(2), 675–712. [https://doi.org/10.1016/s0007-8506\(18\)30216-6](https://doi.org/10.1016/s0007-8506(18)30216-6).
- Nagargoje, A., Kankar, P. K., Jain, P. K., & Tandon, P. (2021). Application of artificial intelligence techniques in incremental forming: A state-of-the-art review. *Journal of Intelligent Manufacturing*. <https://doi.org/10.1007/s10845-021-01868-y>.
- Opritescu, D., & Volk, W. (2015). Automated driving for individualized sheet metal part production - A neural network approach. *Robotics and Computer-Integrated Manufacturing*, 35, 144–150. <https://doi.org/10.1016/j.rcim.2015.03.006>.
- Rossi, G., & Nicholas (2018). Re/Learning the wheel: Methods to utilize neural networks as design tools for doubly curved metal surfaces. *Proc 38th Annu Conf Assoc Comput Aided Des Archit*, 146-155, <https://doi.org/10.52842/conf.acadia.2018.146>.
- Schulman, J., Levine, S., Moritz, P., Jordan, M. I., & Abbeel, P. (2015a). Trust region policy optimization. Proceedings of the 32nd International Conference on Machine Learning, PMLR 37:1889–1897. <https://proceedings.mlr.press/v37/schulman15.html>
- Schulman, J., Moritz, P., Levine, S., Jordan, M. I., & Abbeel, P. (2015b). High-dimensional continuous control using generalized advantage estimation. [arXiv:1707.06347](https://arxiv.org/abs/1707.06347)

- Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. [arXiv:1506.02438](https://arxiv.org/abs/1506.02438)
- Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 1–14.
- Störkle, D., Altmann, P., Möllensiepe, D., Thyssen, L., & Kuhlenkötter, B. (2019). Automated parameterization of local support at every toolpath point in robot-based incremental sheet forming. *Procedia Manufacturing*, 29, 67–73. <https://doi.org/10.1016/j.promfg.2019.02.107>.
- Störkle, D. D., Seim, P., Thyssen, L., & Kuhlenkötter, B. (2016). Machine learning in incremental sheet forming. *47st International Symposium on Robotics*, 2016, 1–7.
- Sutton, R. S., & Barto, A. G. (2017). *Reinforcement Learning: An Introduction (Second edi)*. The MIT Press Cambridge.
- Tanaka, H., Asakawa, N., & Hirao, M. (2005). Development of a forging type rapid prototyping system; Automation of a free forging and metal hammering working. *Journal of Robotics and Mechatronics*, 17(5), 523–528. <https://doi.org/10.20965/jrm.2005.p0523>.
- van der Maaten, L., & Hinton, G. (2008). Visualizing high-dimensional data using t-SNE. *Journal of Machine Learning Research*, 9, 2579–2605.
- van Hasselt, H., Guez, A., & Silver, D. (2016). Deep reinforcement learning with double Q-learning. *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI'16)*, (pp. 2094–2100. <https://doi.org/10.1609/aaai.v30i1.10295>
- Wang, Z., Schaul, T., Hessel, M., van Hasselt, H., Lanctot, M., & de Freitas, N. (2015). Dueling network architectures for deep reinforcement learning. [arXiv.1511.06581](https://arxiv.org/abs/1511.06581)
- Williams, R. J., & Peng, J. (1991). Function optimization using connectionist reinforcement learning algorithms. *Connection Science*, 3(3), 241–268. <https://doi.org/10.1080/09540099108946587>.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.