The International Journal of Advanced Manufacturing Technology (2024) 132:4723–4735
https://doi.org/10.1007/s00170-024-13632-6

ORIGINAL ARTICLE

Springback prediction using point series and deep learning

Yang Bingqian[1] · Yuanyi Zeng[1] · Hai Yang[1] · Mariluz Penalva Oscoz[2] · Mikel Ortiz[2] · Frans Coenen[3] · Anh Nguyen[3]

Received: 13 May 2022 / Accepted: 12 April 2024 / Published online: 25 April 2024
© The Author(s) 2024

Abstract
One of the main challenges that prevent wide adoption of Single Point Incremental Forming (SPIF) is the geometric accuracy of the process resulting primarily from the effect of springback. There are various expedients that can be adopted to address this, but one of the most common is tool path correction. The challenge is then how best to predict springback so as to implement tool path correction. It is established that springback, to a large extent, is related to the geometry of the part to be manufactured. The proposed mechanism uses a novel point series representation to capture local geometries that then form a global bank of geometries for general use. Each point series can then be associated with a predicted springback value generated using deep or machine learning. Experiments are reported using a Long Short Term Memory (LSTM) model coupled with a Multilayer Perception Network (MLP), and a Support Vector Machine (SVM) regression model. A best R2, "Coefficient of Determination", of 0.9181 was obtained indicating that the proposed approach provided a realistic solution to the current limitations of SPIF.

Keywords Single Point Incremental Forming (SPIF) · Springback prediction · Deep learning

List of acronyms

SPIF	Single Point Incremental Forming
LSTM	Long Short Term Memory
MLP	Multilayer Perception Network
SVM	Support Vector Machine
R2	R-square
ANN	Artificial Neural Networks
kNN	K-Nearest Neighbour
RNN	Recurrent Neural Network
BPNN	Back Propagation Neural Networks
CAD	Computer Aided Design
RMSE	Root Mean Squared Error
MSE	Mean Squared Error
MAE	Mean Average Error
GOM	Gesellschaft fur Optische Messtechnik
SGD	Stochastic Gradient Descent
RBF	Radial Based Function

✉ Anh Nguyen
anh.nguyen@liverpool.ac.uk

[1] EITRI (Nanjing Eurosmart Intelligent Technology Research Institute), 9 Chengxin Avenue, Jiangning Development Zone, 21000 Nanjing, China

[2] TECNALIA, Basque Research and Technology Alliance (BRTA), Parque Científico, Parque Científico y Tecnológico de Gipuzkoa, 20009 Donostia-San Sebastián, Spain

[3] Department of Computer Science, The University of Liverpool, Liverpool L693BX, UK

1 Introduction

Single Point Incremental Forming (SPIF) is a non-conventional sheet metal forming technology where a metal sheet is clamped along its perimeter and then deformed locally and progressively under the action of a rounded tool head that follows a numerically controlled tool path. Though it is a slow technology compared to traditional drawing-based processes, it is of benefit with respect to small series, single parts, or prototype productions, because the dedicated tooling cost is decreased and agility is gained. However, tooling suppression also generates geometric accuracy problems, which arguably represents the main challenge for wider adoption in the industry.

Figure 1 shows the different types of geometric errors that can be found in parts produced by SPIF [1]. Springback is due to the elastic recovery of the sheet after deformation and its magnitude depends on the material, the geometry, and the size of the part. Springback is the most critical error affecting the geometric accuracy of SPIF.

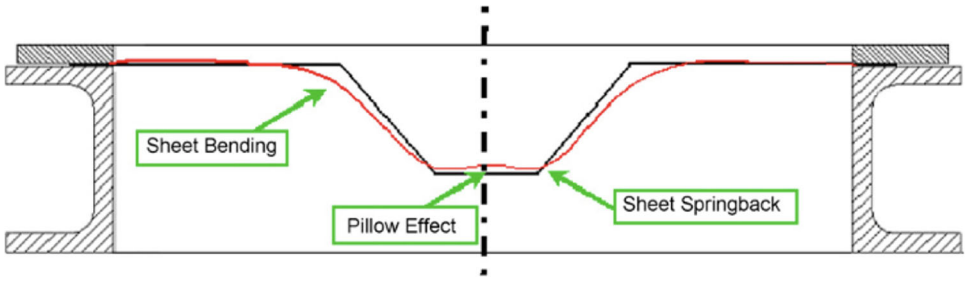


Fig. 1 Typical geometric errors in SPIF

According to [2] the main mechanisms that can be adopted to improve the geometric accuracy of SPIF can be categorised as follows:

- Process parameters optimization
- Multi-stage forming
- Heat-assisted processing
- Process variation
- Tool path correction

Optimization of the process parameters is the most immediate solution. A good understanding of the effect of process parameters on the accuracy is necessary, but the feasibility to control springback deviations through them is limited because they do not act directly on the key influencing factors already mentioned. Moreover, extensive experimentation under the specific conditions of each application is usually needed. Multi-stage forming attempts to mitigate springback through progressive forming stages which have a hardening effect on the material and consequently reduce the elastic recovery. In this case, the main drawback is the significant increase in manufacturing time. Heat-assisted SPIF also contributes to elastic recovery reduction, but its application tends to be limited to materials with low formability at room temperature due to the investment in equipment required. Process variants introduce some kind of means for stiffening the forming area (for example the use of counter supports or dies placed below the sheet). Analogue to heat-assisted SPIF, the equipment cost increases. Finally, tool path correction makes use of a model that predicts the geometric deviations so that, based on the predictions, a corrective strategy is applied to the tool path. This option is commonly used by conventional forming technologies and preserves the original advantages of SPIF in terms of cost and agility. Tool path correction is thus the focus of the work presented in this paper.

The application of AI techniques to predict geometrical deviation (springback) in SPIF has attracted particular attention in recent years. Researchers have experimented with a wide range of AI techniques including the use of Artificial Neural Networks (ANNs) [3], deep and shallow Back Propagation Neural Networks [4] and Regression [5]. An interesting approach was presented in [6] where local geometries were represented as point series. The idea was to establish a bank of local geometries that would form a superset of all the component local geometries that can appear in any part to be manufactured. However, the work presented in [6] defined springback in terms of a set of categorical labels and consequently used traditional machine learning classification techniques, namely k Nearest Neighbour (kNN) classification. The motivation for the work presented in this paper is thus to revisit this earlier work and consider the potential advantage that can be gained if deep learning were adopted to predict real springback values. Specifically, utilising the Long Short Term Memory (LSTM) model, a type of Recurrent Neural Network (RNN), which is well-known for its suitability in time series prediction.

A further disadvantage of the earlier work presented in [6] was that the point series was generated in rather simplistic manner, because of the computing power available at the time. Real values were not used, instead, a set of labels $L = \{negative, level, positive\}$ were used to describe the difference in z value between grid center points and their neighbouring points. A more sophisticated form of point series definition is also presented in this paper. The proposed process is fully described and evaluated.

The rest of this paper is organised as follows. Section 2 presents the related work underpinning the work presented in this paper. The proposed approach is described in Sect. 3 and the experimental set up in Sect. 4. The evaluation results obtained are presented and discussed in Section 5. The paper is concluded in Section 6 with an itemisation of the main find and some suggested avenues for further work.

2 Related work

As noted in the introduction the potential for SPIF has led to a considerable amount of reported research which seeks to address the challenges associated with the technology. Of note is the work of [3–5]. In [3], an ANN was trained using local geometric features represented as 5×5cm height fields and the distances of those local geometries to the edge of the part and to the clamping frame. Data were obtained from a set of panels with slight differences in their design. In [5], the data set was built up from a set of forming experiments by

assigning to each point of the tool path a combination of local geometric features and process parameters. In the reported evaluation 19 different regressors were used (multiple linear, regression trees, SVM, Gaussian process and ensemble learning) to predict springback. A comparison between shallow Back Propagation Neural Networks (BPNN) and deep learners for springback propagation was presented in [4]. In this case, a simple geometry (cone frustum) was studied with the aim of predicting at each of the circles described by the tool path the deviation of the center point and the roundness error.

The above work has focused on several different type of geometrical deviation, in this report, the springback phenomena will be focused on. Previous work [6–8] proposed the idea of predicting springback based on local geometries expressed as point series, and then applying the identified springback values in reverse to correct the tool path was considered. The point series were generated by dividing the shape to be manufactured into a set of equal size grid squares. Each grid square was defined by a set of $[x, y, z]$ coordinates referenced to an origin. The desired point series were then generated by considering the neighbouring grid squares to each current grid square and consecutively determining the difference in z value. The proposed approach was shown to work well and a similar idea is adopted with respect to the work presented in this paper. The distinction between the work in [6–8] presented here was that in the earlier work, the difference in z value was expressed as a set of nominal values. A further distinction is that springback prediction was conducted in terms of classification; this was because of the limited processing power available to the researchers at the time. Classification is a form of supervised learning. A classifier is trained using a pre-labelled data set. The particular classification model used was the k Nearest Neighbour (kNN) model coupled with Dynamic Time Warping [10] for time series similarity comparison. However, the kNN methods is an instance-based learning algorithm used for classification and regression problems. kNN classifies or predicts based on the labels of the nearest neighbors in the feature space. Considering that historical deformation information is also an important factor when springback happens, kNN has a limited ability in terms of describing long-term dependencies in point series. In this paper, instead of using the classification method kNN, the LSTM neural network in the deep learning field has been used to predict springback. LSTM is designed to handle sequential data and is particularly effective for tasks that involve considering context and temporal dependencies. The point series generation method is also been improved; the real value of springback has been used as a label other than categorical label $L = \{L_1, L_2, L_3, L_4, L_5\}$ which represents different levels of springback value. The detailed method was described in Section 3.1.

3 Springback prediction using point series and deep learning

This section presents the proposed approach to predicting springback so that corrections can be applied to the definition of a proposed shape. The fundamental idea was to build a springback prediction model using deep learning technology. Once learned the model can be used to predict springback given a new shape. To generate the model training data was required; a data set $D_{train} = \{T_1, T_2, \dots\}$ where each $T_i \in D_{train}$ is a fragment of shape geometry that has been labeled with an identified springback value e. There are a variety of ways in which fragments of geometry can be represented. As noted earlier, in [6] point series was used; a similar approach was used with respect to the work presented in this paper. Thus, each $T_i \in D_{train}$ comprises a tuple of the form $\langle P, e \rangle$, where P is a point series $P = [p_1, p_2, \dots]$ and p_j is a point value in the point series, and e is a springback value. We generated D_{train} using previously manufactured pieces for which before and after point clouds were available. The process for generating D_{train} is presented in Sub-section 3.1 below. Once D_{train} has been generated it can be used to create a springback model. The process for this is described in Sub-section 3.2.

3.1 Training data generation

In this sub-section, the process for generating the desired training data is presented. The input comprises two-point clouds F_{in} and F_{out} such that: F_{in} describes the piece to be manufactured, generated from the Computer-Aided Design (CAD) software package originally used to specify the piece to be manufactured; and F_{out} describes the resulting manufactured piece, generated using 3-D optical photogrammetry. The two-point clouds were aligned as part of the photogrammetry process. The output was the set $D_{train} = \{T_1, T_2, ...\}$ as described above. The point series were generated by superimposing a grid over the input point cloud. Figure 2 shows the overall process. From the figure, it can be seen that the process comprises three sub-processes: (i) grid representation, (ii) error calculation, and (iii) point series generation. Each is discussed in further detail in the remainder of this subsection.

3.1.1 Grid generation

From Fig. 2, the first sub-process is Grid Generation. Grid representations are commonly used to describing 2D surfaces, but can equally be applied to 3D surfaces by considering only the x and y dimensions [9]. Given F_{in} and F_{out} we wish to generate two corresponding grids G_{in} and G_{out}. Each set of grid squares G (either G_{in} and G_{out}) is of

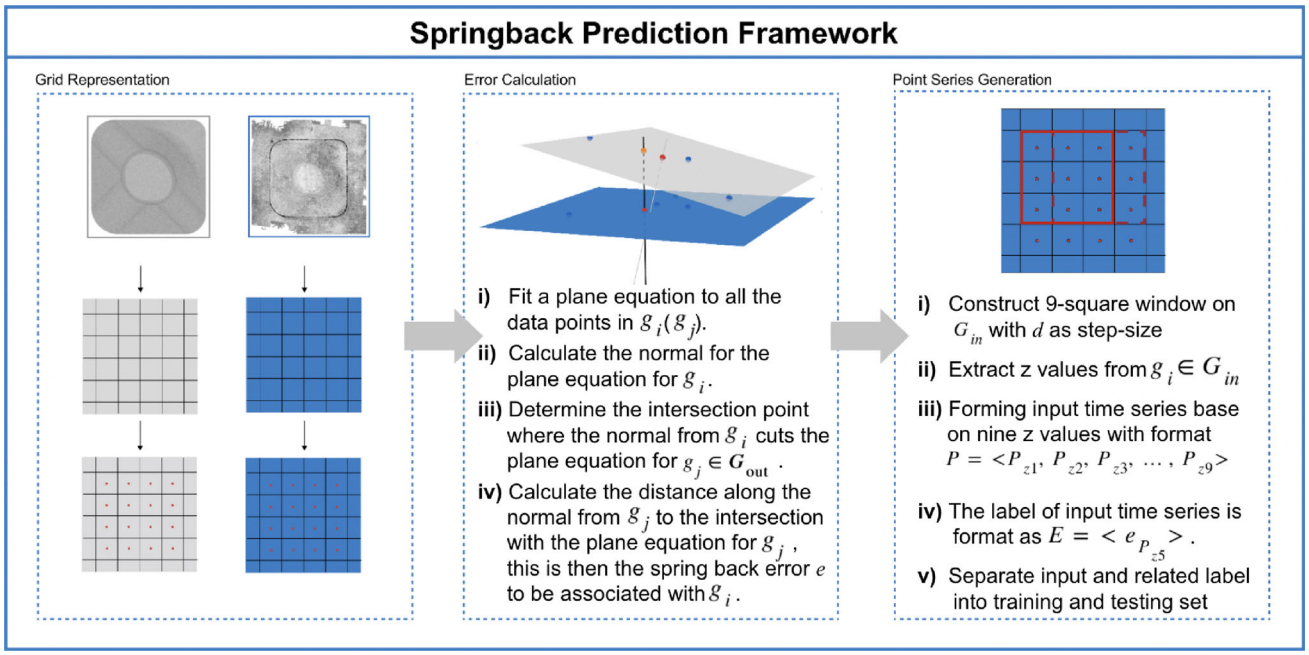


Fig. 2 Schematic showing springback prediction training data generation process

the form $\{g_1, g_2, \dots\}$ where $g_i \in G$ comprises a set of grid coordinates $[x, y, z]$ such that x and y are the coordinates of the grid centre point and z the associated z-coordinate value. The geometric centre of F_{in} and F_{out}, in plan view, was considered to be the x-y origin for the grid, with an associated z-value of 0, corresponding to the minimum z-value in the given point clouds (so that all z-vales would be positive). Once the origin was established each grid was then generated simply by dividing the space, using a predefined grid size d, into a set of non-overlapping grid squares, each measuring $d \times d$ and each defined by the x-y coordinates of its centre point. The z-value for each grid square was calculated by averaging all the z-values, from F_{in} or F_{out} as appropriate, contained in the grid square. Figure 3 shows an example grid. Similar approaches were used in [9] and [10].

3.1.2 Error calculation

The second sub-process given in Fig. 2 is Error Calculation; the determination of the springback value for each grid square $g_i \in G_{in}$. The error values were stored in a set $E = \{e_1, e_2, \dots\}$ such that there was a one-to-one correspondence with G_{in}. The error values were derived by determining the distance along the normal from the center point of each grid square $g_i \in G_{in}$ to where it cuts G_{out}.

Given a grid square $g_i \in G_{in}$ and a corresponding grid square $g_j \in G_{out}$ the error is calculated by following the process outlined in [7, 11, 12]:

1. Fit a plane equation to all the data points in g_i (g_j).
2. Calculate the normal for the plane equation for g_i.
3. Determine the intersection point where the normal from g_i cuts the plane equation for $g_j \in G_{out}$.
4. Calculate the distance along the normal from g_i to the intersection with the plane equation for g_j, this is then the springback error e to be associated with g_i.

For each $g_i \in G_{in}$ and $g_j \in G_{out}$ pairing; the coefficients, $\widehat{\boldsymbol{\beta}}$, of the associated plane equation can be generated using

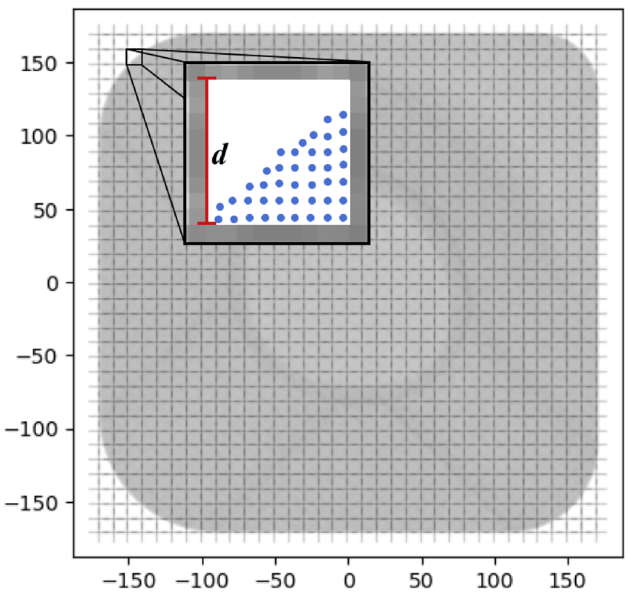


Fig. 3 Example grid

the Ordinary Least Square (OLS) method as shown in Eqs. 1 to 4. The dependent variable is the z-coordinate, **z**, of the points in g_{in} (g_{out}) and the independent variables are the x- and y-coordinates, [**x**, **y**].

$$\mathbf{z} = [\mathbf{x}, \mathbf{y}]\boldsymbol{\beta} + \boldsymbol{\varepsilon} \tag{1}$$

$$\begin{aligned} RSS(\boldsymbol{\beta}) &= \boldsymbol{\varepsilon}^\top \boldsymbol{\varepsilon} \\ &= (\mathbf{z} - [\mathbf{x}, \mathbf{y}]\boldsymbol{\beta})^\top (\mathbf{z} - [\mathbf{x}, \mathbf{y}]\boldsymbol{\beta}) \\ &= \mathbf{z}^\top \mathbf{z} - \boldsymbol{\beta}^\top [\mathbf{x}, \mathbf{y}]^\top \mathbf{z} - \mathbf{z}^\top [\mathbf{x}, \mathbf{y}]\boldsymbol{\beta} \\ &\quad + \boldsymbol{\beta}^\top [\mathbf{x}, \mathbf{y}]^\top [\mathbf{x}, \mathbf{y}]\boldsymbol{\beta} \rightarrow \min_{\beta_0, \beta_1, \beta_2} \end{aligned} \tag{2}$$

$$\frac{\partial RSS(\widehat{\boldsymbol{\beta}})}{\partial \widehat{\boldsymbol{\beta}}} = -2[\mathbf{x}, \mathbf{y}]^\top \mathbf{z} + 2[\mathbf{x}, \mathbf{y}]^\top [\mathbf{x}, \mathbf{y}]\widehat{\boldsymbol{\beta}} = 0 \tag{3}$$

$$\widehat{\boldsymbol{\beta}} = \left([\mathbf{x}, \mathbf{y}]^\top [\mathbf{x}, \mathbf{y}]\right)^{-1} [\mathbf{x}, \mathbf{y}]^\top \mathbf{z} = [b_0, b_1, b_2] \tag{4}$$

Once the Coefficients have been obtained, the plane equation can be written as shown in Eq. 5 and the normal vector of the plane written as shown in Eq. 6.

$$\mathbf{Z} = b_1 * \mathbf{X} + b_2 * \mathbf{Y} \tag{5}$$

$$[b_1, b_2, -1] \tag{6}$$

Given the Plane equation of each grid square $g_i \in G_{in}$ and each grid square $g_j \in G_{out}$, the distance along the normal from the centre point of each g_i to where it intersects with G_{out} can be calculated. The mechanism for achieving this is shown in Fig. 4 [13]. The intersection point in G_{out} can be identified based on Eq. 7, where $intersect_x$, $intersect_y$ and $intersect_z$ are the x-y-z coordinates for the intersection point; and g_{j_x}, g_{j_y} and g_{j_z} are the x-y-z coordinates for grid square $g_j \in G_{out}$.

$$\frac{(intersect_x - g_{j_x})}{b_1} = \frac{intersect_y - g_{j_x}}{b_2} = \frac{intersect_z - g_{j_z}}{-1} \tag{7}$$

The distance along the normal from a grid square $g_i \in G_{in}$ and the intersection point $intersection$, the springback, is measured using Eq. 8. The $error$ term is the springback value of this grid square.

$$error = \sqrt{(intersect_x - g_{i_x})^2 + (intersect_y - g_{i_y})^2 + (intersect_z - g_{i_z})^2} \tag{8}$$

The direction (sign) of the springback error is defined by the z-coordinate value of the intersect point, $intersect_z$, and the z-coordinate value of the current grid square $g_{i_z} \in C_{in}$. This is calculated as shown in Eq. 9.

$$\text{springback direction} = \begin{cases} intersect_z \geq g_{i_z}, & +ve \\ \text{otherwise} & -ve \end{cases} \tag{9}$$

3.1.3 Point series generation

The final sub-process in the training data generation process is point series generation. The idea is to capture the local geometry surrounding each $g_i \in G_{in}$ in terms of the difference in z value between g_i and each of its neighbouring grid squares as shown in Fig. 5. Each $g_i \in G_{in}$ was represented as a labeled point series $P = \langle P, e \rangle = \langle [p_1, p_2, \ldots, p_n], e \rangle$ where p_j is the difference in z value between grid square g_i and its neighbour g_j, and e is the corresponding springback value in E. There are a variety of configurations for such point series, but the simplest is to consider the eight immediate neighbours of each grid square in G_{in} (five at edges, and three at corners) as shown in Fig. 5. The order in which the point series are generated is not significant as long as it is done in a consistent manner.

Once the set of point series $\{P_1, P_2, \ldots\}$, and the set of errors $\{e_1, e_2, \ldots\}$ has been calculated the point series and errors can be incorporated into the training set $D_{train} = \{T_1, T_2, \ldots\}$ where each $T_i = \langle P_i, e_i \rangle$. The next stage is prediction model generation.

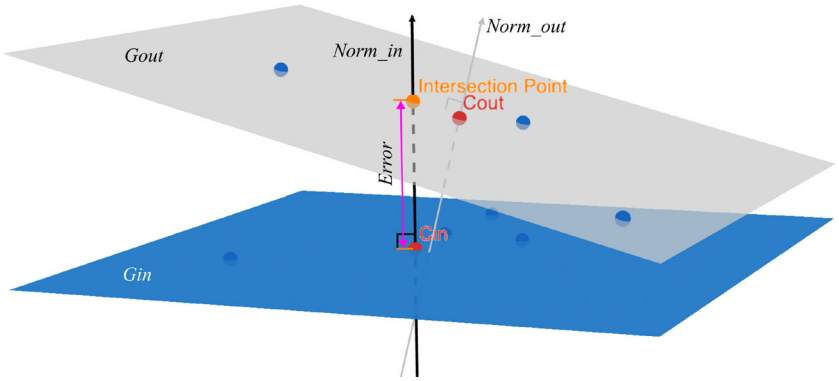


Fig. 4 Intersection and error calculation [13]

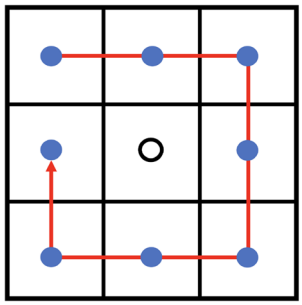


Fig. 5 Point series generation

3.2 Prediction

Once D_{train} has been generated, as described in the previous sub-section, we use it to train a learning model that can predict springback errors for previously unseen pieces. For the work presented in this paper two prediction models were considered. A Long Short Term Memory (LSTM) model [14] with a Multilayer Perception Network (MLP) [15]. For comparison purposes, as reported in the evaluation presented in the following section, a Support Vector Machine (SVM) regression model [16] was also considered.

LSTM is a popular Recurrent Neural Network (RNN) introduced by Hochreiter and Schmidhuber [14] in 1997. It has achieved great success in many sequential data prediction tasks, for instance, handwritten text recognition [17] and speech recognition [18]. Though LSTM is more commonly used in time series data to learn long-term dependencies, the spatial dependencies in an image feature space can also be learned [19].

When generating an LSTM model, for each time period t, the LSTM gets one input x_t, and computes a new cell state c_t and a hidden state h_t based on the new input and memory from the previous time period as shown in Eq. 10 where i refers to the input gate, f stands for the forget gate, o is the output gate, and c refers to the cell activation vectors; $ReLU$ stands for Rectified Linear Unit non-linearity function [20], ϕ represent the hyperbolic tangent non-linearity function, and $\circ$ stands for to element-wise multiplication.

$$
\begin{aligned}
i_t &= ReLU(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \\
f_t &= ReLU(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \\
o_t &= ReLU(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \\
g_t &= \phi(W_{xg}x_t + W_{hg}h_{t-1} + b_g) \\
c_t &= f_t \circ c_{t-1} + i_t \circ g_t \\
h_t &= o_t \circ \phi(c_t)
\end{aligned} \tag{10}
$$

Figure 6 shows an overview of the adopted LSTM approach. From the figure it can be seen that the architecture comprised the following parts: (i) an LSTM layer with 128 internal units, (ii) a dense (fully connected) layer with 256 nodes, (iii) another dense layer with 64 nodes, and (iv) a final dense layer with a single node to predict the springback error. The input was the tuples within the training data set, $\langle P, e \rangle$. The LSTM layers then learn these vectors based on the spatial relationships that feature within them. The LSTM layers output the feature vectors from the input data. After the LSTM layers, two fully connected layers are implemented. Finally, a fully connected layer with one neuron is used to act as the "predictor" for the network.

4 Evaluation setup

This section presents the evaluation setup of the proposed time series-based approach. For the evaluation three springback data sets were used, simply designated as evaluation sets 1, 2, and 3. Further detail concerning these data sets is presented in Section 4.1. A three-fold evaluation strategy was adopted whereby one of the evaluation data sets was used for training and the other two for testing. Six sets of results were therefore obtained.

The performance metrics used were: Root Mean Squared Error (RMSE), (ii) Mean Squared Error (MSE), (iii) Mean Average Error (MAE), and (iv) R2. Some further detail concerning these metrics is given in Section 4.2. Details concerning the implementation of the adopted LSTM-MLP in Section 4.3. The results obtained are discussed in the following section, Section 5.

4.1 Evaluation data set

For the evaluation of the proposed approach three manufactured pieces were used, Piece 1, Piece2, and Piece 3; from the same CAD specification. Figure 7 gives visualisations for the C_{in} (CAD) and three C_{out} point clouds. The C_{out} point clouds are generated use Gesellschaft fur Optische Messtechnik(GOM), an optical measuring system. From the figure, it can be seen that the C_{out} clouds featured a certain amount of missing data. This was not considered significant because we were only interested in the area within the operational boundaries. Hence local geometries that featured an insufficient number of points, outside these boundaries, were ignored. Note that the tool head used to manufacture the three pieces measured 20 mm in diameter.

A 5-axis tool path was programmed to fabricate the three parts using the parameters listed in Table 1. Due to the limited formability of titanium at room temperature, the parts were fabricated in a TECNALIA gantry machine equipped with a furnace to keep the sheet at a homogeneous temperature during forming (Fig. 8). Based on previous analysis from [21] the forming temperature was set to 675 °C and the heating up and cooling down cycles were defined. Boron Nitride was

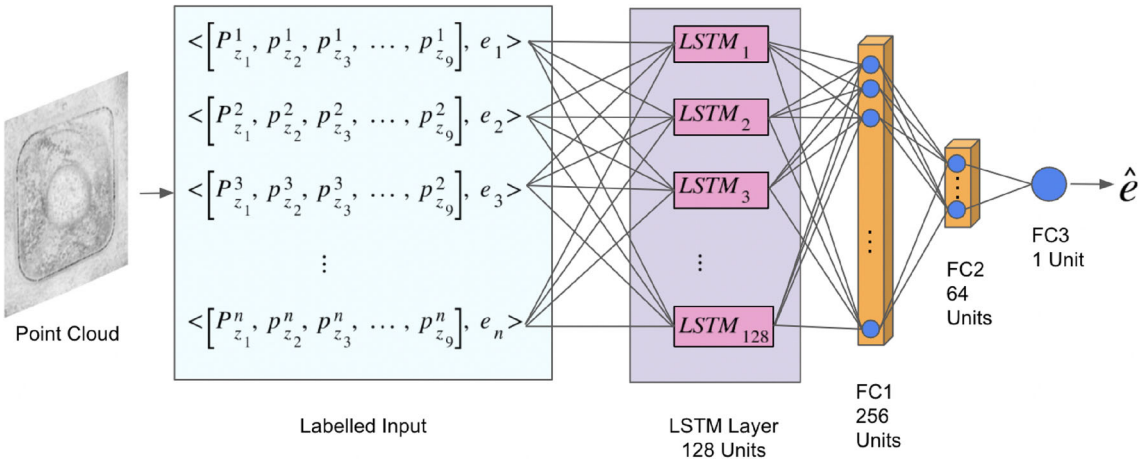
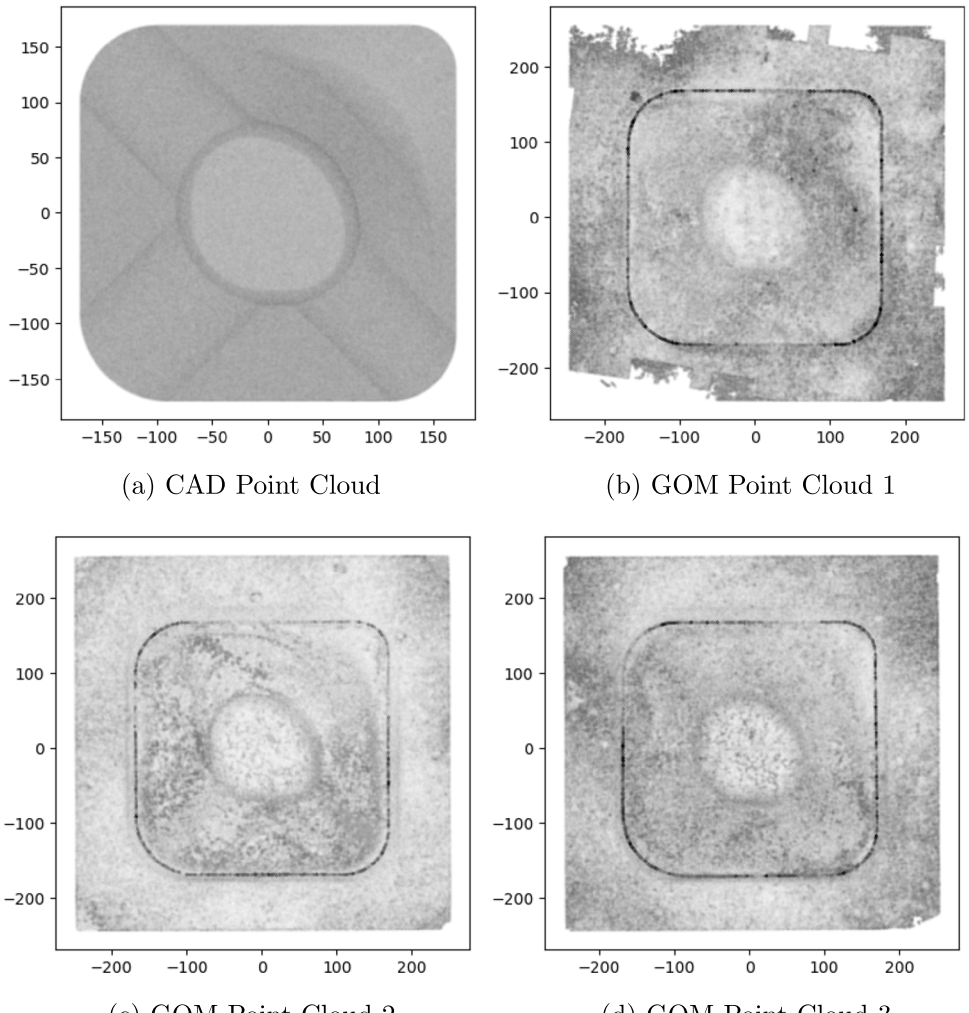


Fig. 6 LSTM network architecture



(a) CAD Point Cloud (b) GOM Point Cloud 1

(c) GOM Point Cloud 2 (d) GOM Point Cloud 3

Fig. 7 The C_{in} (CAD) and C_{out} (GOM) point clouds used for the evaluation

Table 1 Tool path parameters

Forming Tool	R50×r11mm Ceramic Wheel
Traverse Speed	1000mm/min
Step Down	0.29mm (0.5mm at lowest levels)
Tilt Angle	30°C

applied to the sheets as a lubricant. The produced parts were measured using the GOM ATOS optical 3D system. The point clouds obtained were aligned with the original CAD for error calculation by using the GEOMATIC control software by 3D SYSTEMS.

4.2 Evaluation metrics

For the evaluation, we compare the springback values predicted by our model with known springback values. In other words, we had a set of n expected values $E = \{e_1, e_2, \ldots, e_n\}$ and a set of n predicted values $\hat{E} = \{\hat{e}_1, \hat{e}_2, \ldots, \hat{e}_n\}$, we wished to determine the correspondence between the two as a result of applying the proposed springback prediction model. The following performance metrics were used: (i) Mean Absolute Error (MAE), (ii) Mean Square Error (MSE), and (iii) Root Mean Square Error (RMSE), and (iv) R2.

MAE is calculated as shown in Eq. 11. The $|x|$ notation used in the equation should be interpreted as indicating the *absolute* value of x (the magnitude of x). From the equation, it can be seen that MAE takes into account the the magnitude of the error only, not the direction; in many cases, Mean Absolute Error is a good indicator of performance:

$$e_{mae} = \frac{\sum_{i=1}^{i=n}(|e_i - \hat{e}_i|)}{n} \tag{11}$$

MSE is calculated as shown in Eq. 12. From the equation, it can be seen that MSE accentuates larger errors. However, a disadvantage is that the error value is not expressed using the same units as used for E and P.

$$e_{mse} = \frac{\sum_{i=1}^{i=n}(e_i - \hat{e}_i)^2}{n} \tag{12}$$

RMSE is calculated as shown in Eq. 13. RMSE is similar to MSE but with the error value expressed using the same units as used for E and P (mm with respect to the evaluation reported here):

$$e_e = \sqrt{P\frac{\sum_{i=1}^{i=n}(e_i - \hat{e}_i)^2}{n}} \tag{13}$$

The R2 measure, also known as the "Coefficient of Determination", is calculated as shown in Eq. 14. In the equation, the notation $\bar{\hat{E}}$ indicates the mean of the predicted values. R2 is closely related to RMSE. With respect to MAE, MSE, and RMSE the lower the value the better our model. R2 indicates how well the predictions made by our model are related to the expected values, so it is a measure of the quality of the model rather than the predictions themselves. An R2 value will be between 0 and 1; 0 indicates that the model does not fit the training data at all, and 1 indicates that the model fits the training data perfectly. Thus, the higher the R2 value the better the model.

$$e_{r2} = 1 - \frac{\sum_{i=1}^{i=n}(e_i - \hat{e}_i)}{\sum_{i=1}^{i=n}(e_1 - \bar{\hat{e}})} \tag{14}$$

4.3 Implementation detail

The operations of the LSTM-MLP model were compared with an SVM Regression model. Both were implemented in Python using the Keras Python package. As noted earlier the LSTM-MLP architecture comprised: (i) an LSTM layer with 128 internal units, (ii) a dense (fully connected) layer with 256 nodes, (iii) another dense layer with 64 nodes, and (iv) a final dense layer with a single node to predict the springback error. The LSTM layer was included to capture



(a) Overview (a) Detail

Fig. 8 Equipment used for SPIF fabrication of evaluation samples

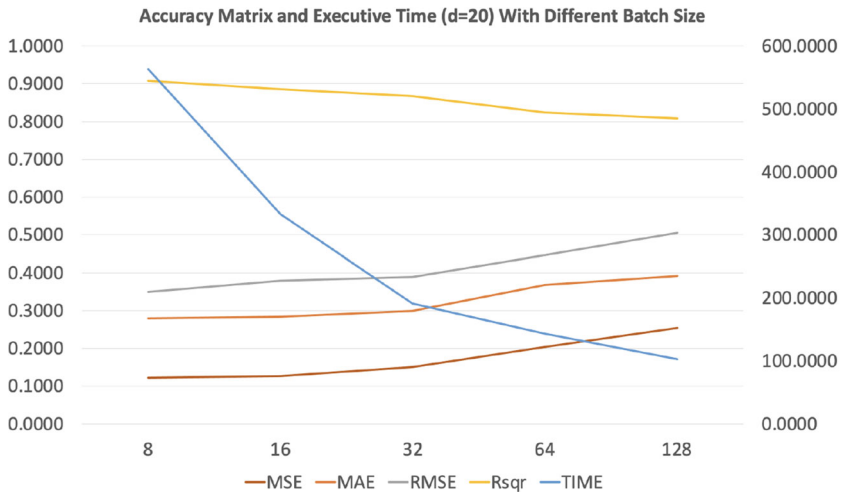


Fig. 9 Accuracy Matrix and Execution time for $d = 20$mm

long-term dependencies in sequential data, while the Dense layers were used to learn high-dimensional feature representations. The choice of 128 LSTM units and Dense layers with 256 and 64 nodes struck a balance between complexity and computational requirements. To accelerate convergence and reduce oscillations during training the Stochastic Gradient Descent (SGD) optimizer was used with a momentum of $m = 0.01$; together with a learning rate of $\alpha = 0.001$ and an patience epoch size of $e = 50$. Thus, the value for α was updated, whenever 50 epochs was reached and the validation loss had not improved; α was reduced by half until a minimum value $\alpha = 0.00001$ was arrived. A higher learning rate can often speed up convergence but may risk overshooting, thus, a dynamic adjustment mechanism has been adopted here. The input data set was split into training and validation subsets using a ratio of $8 : 2$. Also, to further accelerate the training process, the data was grouped into batches.

For the SVM regression model, a cost parameter value of $c = 10$ was used, a margin of tolerance of $\epsilon = 0.1$, and a gamma parameter value of $\gamma = 0.001$. The cost parameter defines the extent to which an SVM model is allowed to "bend" to fit the training data. The gamma parameter defines the "influence" of a single training example, the lower the value the greater the influence. The popular Radial-Based Function (RBF) Kernel [22] was used; RBF measures the similarity between each time series to allow linearly separable.

5 Results and discussion

The objectives of the evaluation were as follows:

1. To determine the most appropriate batch size for the advocated LSTM-MLP springback prediction model. A range of batch sizes were considered, {8, 16, 32, 64, 128}.
2. To compare the operation of the advocated LSTM-MLP springback prediction model with an alternative model, namely an SVM regression model.

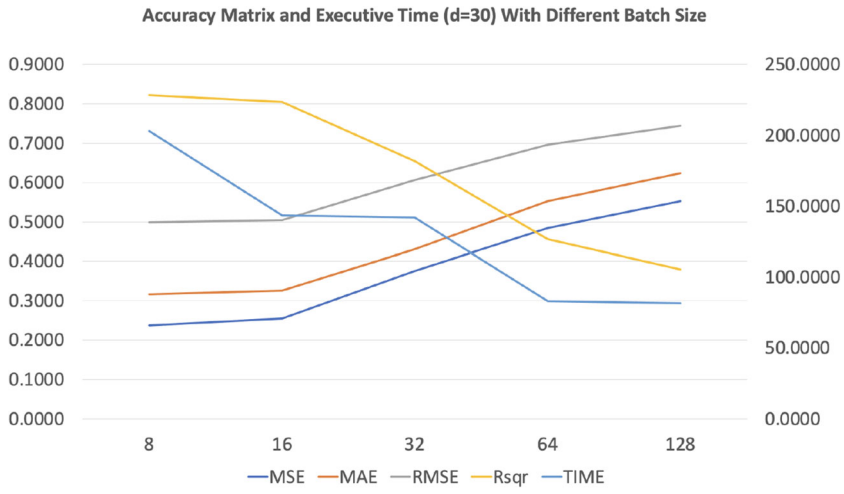


Fig. 10 Accuracy matrix and execution time for $d = 30$mm

3. To investigate the effect of the grid size d on prediction accuracy. A range of grid sizes were considered, {2, 5, 10, 15, 20, 30, 40, 50}mm.

Each is discussed in further detail in the following three subsections, Sections 5.1, 5.2 and 5.3.

5.1 Evaluation results—batch size

To determine the optimum LSTM-MLP batch size in terms of accuracy and training time, as noted above, a range of batch sizes were experimented with, {8, 16, 32, 64, 128}. Experiments were conducted using $d = 20$ and $d = 30$. Of course, with the same data set, the smaller the batch size the greater the training time. The results obtained are shown in Figs. 9 and 10 for $d = 20$mm and $d = 30$mm respectively. From figures, as batch size increases, the MSE, MAE, and RMSE are showing an increasing trend, whereas the Rsqr percentage is showing a decreasing trend. The trends are all leading to the conclusion that, as batch size increases, the model explainability on data is decreasing. From Fig. 9 ($d = 20$mm), a batch size of 32 can be argued to provide the best compromise between accuracy and training time. However, as shown in Fig. 10 ($d = 30$mm), there is a pivot decrease in accuracy at a batch size of 32. Thus, a batch size of 16 can be argued to be the best choice when $d = 30$mm. In conclusion, considering the trade-off between accuracy and training time, a batch size of 16 was considered to be the most appropriate. This was thus the batch size used with respect to the further evaluation reported on in the following two sub-sections.

5.2 Evaluation results—grid size

The average results obtained, from experiments using the six combinations of the three evaluation data sets, and a range of different values for d, are presented in Tables 2 and 3. Table 2 presents the average results obtained using the LSTM-MLP model and Table 3 the average results obtained using the SVM model. From the table, it can be seen that for the LSTM-MLP approach, the best results were obtained at low grid sizes (d less than 10mm) and the SVM regression approach when $d = 30$mm. However, there is a trade-off between accuracy and model generation time.

Table 2 Average evaluation results using LSTM-MLP model (best results in bold font)

Grid Size	MAE	MSE	RMSE	R2
50	0.4423	0.4391	0.6579	0.6940
40	0.3070	0.2318	0.4703	0.8147
30	0.3274	0.2313	0.4807	0.8286
20	0.3028	0.1640	0.4043	0.8751
15	0.3129	0.1664	0.4071	0.8750
10	**0.3002**	**0.1596**	0.3981	0.8921
5	0.3067	0.1580	**0.3963**	0.9140
2	0.3341	0.1856	0.4283	**0.9181**

Table 3 Average evaluation results using SVM model (best results in bold font)

Grid Size	MAE	MSE	RMSE	R2
50	0.3907	0.3529	0.5934	0.7526
40	0.3058	0.2164	0.4598	0.8265
30	0.**2877**	**0.1366**	**0.3668**	**0.8997**
20	0.2925	0.1452	0.3796	0.8899
15	0.3239	0.1784	0.4215	0.8668
10	0.3592	0.2206	0.4688	0.8518
5	0.4569	0.3637	0.6005	0.8036
2	0.6622	0.8063	0.8955	0.6452

Figure 11 shows the average training time, for different grid sizes, using the advocated LSTM-MLP approach. The negative exponential relationship between grid size and training time can be seen. As the grid size decreases the number of individual geometries increases in an exponential manner, and this raises a computation complexity issue. If the model only needs to be generated once this is not an issue. If the model needs to be repeatedly generated this may become significant; but this is not anticipated to be the case. Comparing the results presented in Table 2 and Fig. 11 it can be argued that a grid size of between $d = 5$mm and $d = 10$mm presents the best trade-off between training time and accuracy when using the LSTM-MLP model.

5.3 Evaluation results—LSTM-MLP versus SVM Regression

The accuracy values obtained using LSTM-MLP and SVM Regression were given in Tables 2 and 3 respectively. The same results are given in graphical form in Fig. 12. Inspection of the results indicates that the best MAE, MSE, and RMSE values were obtained using the SVM regression model. The best R2 result was obtained using the LSTM model, recall that R2 measures the goodness of the model rather than prediction accuracy. From the tables and figure it can be seen that best results were obtained using SVM regression and a grid size of $d-30$. However, when using the SVM regression model, as we move away from the $d = 30$ grid size, in either direction, the performance deteriorates quite rapidly. When the grid size increases, the area represented becomes too large for effective springback predictions to be made. The the rea-

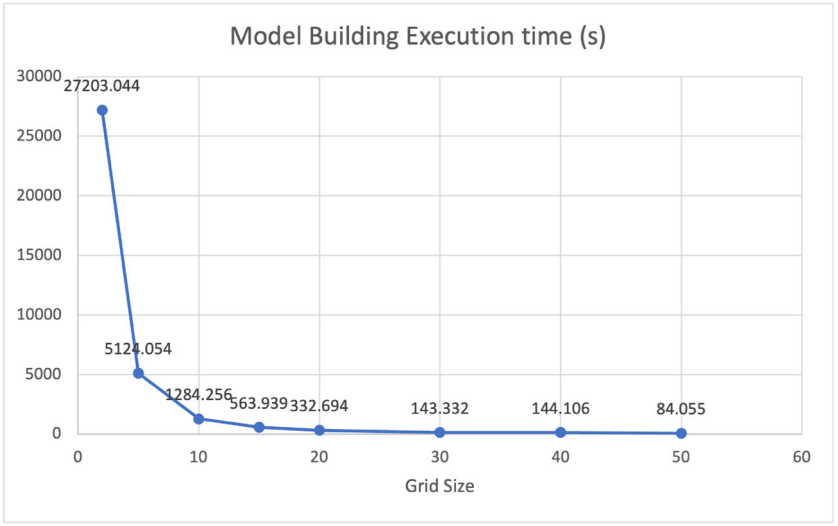


Fig. 11 Run time, using different grid sizes, for LSTM-MLP model generation

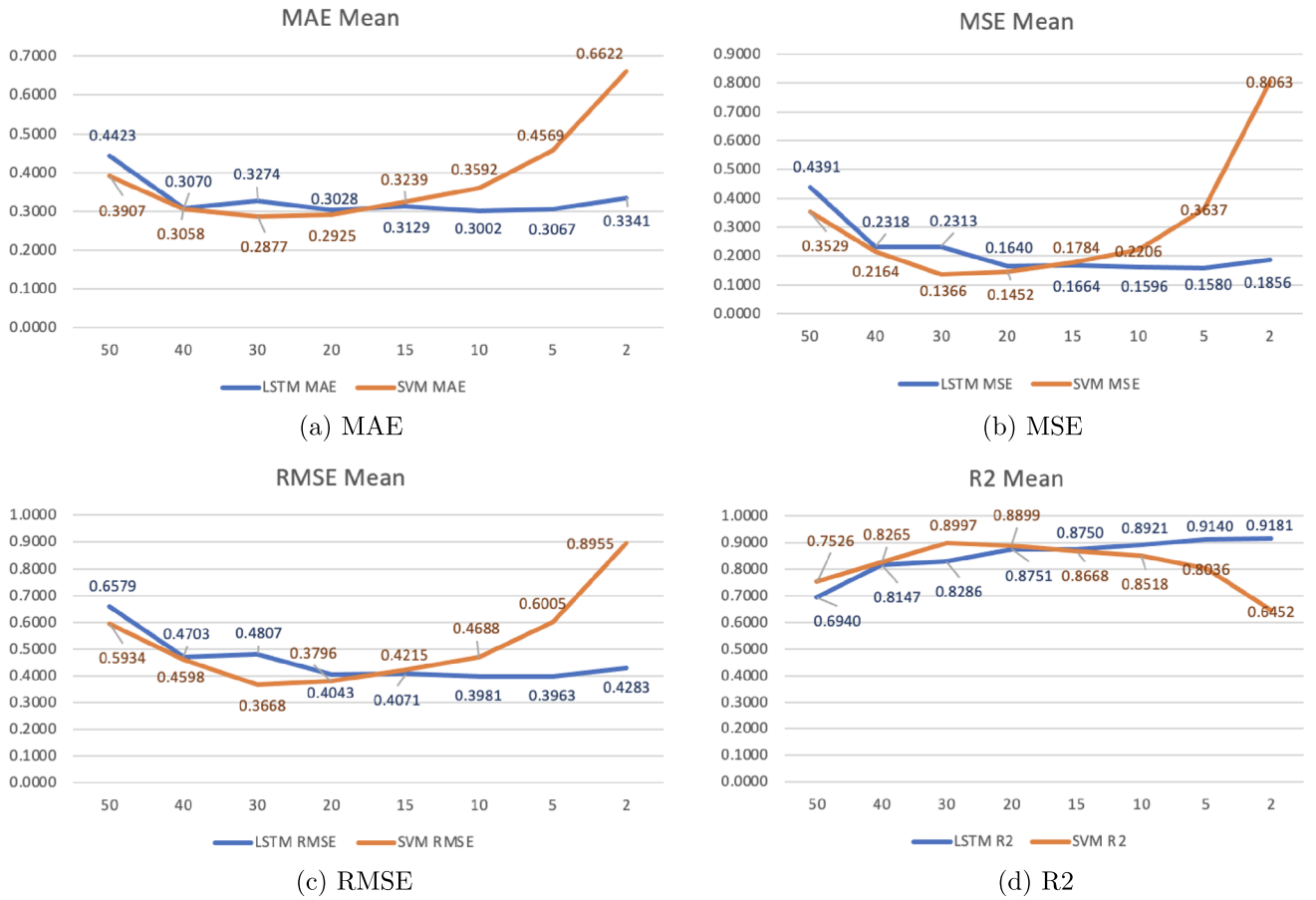


Fig. 12 Combined evaluation results

son behind SVM's better performance is due to the smaller data size provided($d >= 30$), the number of input point series available for training has decreased, SVM compared to LSTM can demonstrate better generalisation and adaptation capabilities when limited samples are provided. LSTM often requires a large amount of labeled sequential data for training and capturing long-term dependencies. Therefore, though SVM outperforms LSTM in $d = 30$ situation it is inappropriate to conclude SVM outperforms LSTM in general.

When the grid size decreases the area represented becomes too small for effective springback predictions to be made and the performance of SVM is poor compared to LSTM when data size increases ($d < 30$). This is because as data complexity increases, LSTM that designed to handle nonlinear and time-dependent problems can capture more feature information and, thus, outperform SVM in all perspectives. Moreover, for the LSTM model, at lower grid sizes, less than 20 mm, the model becomes relatively stable, and the small fluctuations in performance are acceptable in return for stability (the specific grid size is not as important as in the case of the SVM model).

From the application perspective, when considering the purpose of the model, the springback prediction model is built to assist the manufacturing process where the tool size and metal sheet shape are not always the same. The SVM model though shows lower MAE, MSE, and RMSE values just at $d = 30$ grid size, the LSTM model can provide better flexibility as it can automatically adjust the model through the process of backpropagation and gradient updates to adapt these variations. This can be beneficial during manufacturing, especially for model further fine-tuning if the sheet shape has been adjusted. In contrast, SVM requires manual tuning of hyperparameters, which may need to be readjusted for different datasets and problems, resulting in less adaptive behavior.

Furthermore, from a model perspective, the LSTM model is capable of handling sequential data. In the prediction of springback in metal sheet forming, it is necessary to consider the temporal and continuous nature of the material, as the historical deformation information is crucial for predicting springback. The SVM, on the other hand, is suitable for processing static and non-sequential data, making it challenging to effectively capture the temporal dependencies in a sequence. Furthermore, LSTM has the ability to adaptively learn and adjust model parameters based on the characteristics of the input data. In the prediction of springback in metal sheet forming, different materials and operating conditions may show distinct properties and springback patterns. Thus, LSTM model is more appropriate if trading off between strengths and weaknesses discussed.

6 Conclusion

This paper has presented an approach to SPIF springback prediction using a novel point series local geometry representation founded on a grid conceptualisation defined by a given grid size d. The intuition was that the local geometries could be combined in a variety of manners to describe any part to be manufactured. The proposed point series representation was founded on that used in [6], but using real point values as opposed to categorical point values. The proposed representation was used to generate springback prediction models. Two mechanisms were considered whereby such models could be produced: LSTM-MLP and SVM Regression. The reported evaluation firstly indicated that a batch size of 16 presented the best trade-off between training time and accuracy with respect to LSTM-MLP model generation. Secondly, it was found that although the SVM regression model produced the best overall accuracy, it was relatively unstable compared to the LSTM-MLP model where acceptable accuracies were obtained over a range of grid sizes. The best value for d was found to be 10mm (half the tool head size). Overall, the work presented in this paper indicates that the fundamental idea of representing local geometries as time series, local geometries that can be combined with respect to any shape, and which can then have a predicted springback value associated with them, generated using an appropriate deep learning or machine learning model, may serve to allow SPIF to be used in a wider commercial context. For instance, if a new theoretical CAD cloud point has been constructed. The point cloud can be processed and fed to the trained LSTM model. The model will make a prediction and give a potential springback value that might take place. Then engineers are able to adjust the toolpath based on the prediction result to mitigate springback effect during the next round of metal sheet formation.

Furthermore, the step can be repeated, if the shape of the first-round metal sheet does not meet the standard. The cloud point of this first-round metal sheet can then be passed into the model. Another set of springback predictions can be made, the toolpath can be further adjusted for the next round of forms. In this way, by correcting the toolpath step by step, the shape will become closer to the theoretical standard. By utilising springback prediction model during the forming process, we can save half of the forming repetitions and time.

Although this study has achieved significant findings in the prediction of springback using point series and LSTM model, there are several potential research directions and areas for improvement that can further expand and deepen the research in this field. As the LSTM model is flexible in taking new shapes and datasets, to further improve

the accuracy and reliability of springback prediction, it is worth considering expanding the size and diversity of the dataset. By collecting data from a wider range of samples and different materials, the generalizability of the model can be enhanced, leading to improved prediction performance under various processes and conditions. Moreover, the LSTM model used in this research has demonstrated excellent performance in springback prediction. However, there are other deep learning models and algorithms that can be explored and compared. For instance, graph neural networks(GNN) which could offer additional benefits by effectively capturing the inherent topology and inter-connectedness of the springback phenomenon.

Declarations

Conflict of interest The authors declare no competing interests.

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

1. Micari F, Ambrogio G, Filice L (2007) Shape and dimensional accuracy in single point incremental forming: state of the art and future trends. J Mater Process Technol 191(1–3):390–395
2. Harfoush A, Haapala KR, Tabei A (2021) Application of artificial intelligence in incremental sheet metal forming: a review. Procedia Manuf 53:606–617
3. Zwierzycki M, Nicholas P, Ramsgaard Thomsen M (2018) Localised and learnt applications of machine learning for robotic incremental sheet forming. In: Humanizing digital reality. Springer, pp 373–382
4. Akrichi S, Abbassi A, Abid S, Ben Yahia N (2019) Roundness and positioning deviation prediction in single point incremental forming using deep learning approaches. Adv Mech Eng 11(7):1687814019864465
5. Möllensiep D, Kulessa P, Thyssen L, Kuhlenkötter B (2020) Regression-based compensation of part inaccuracies in incremental sheet forming at elevated temperatures. Int J Adv Manuf Tech 109(7):1917–1928
6. M. S. Khan, F. Coenen, C. Dixon, S. E. Salhi, M. Penalva, A. Rivero, (2014) An intelligent process model: predicting springback in single point incremental forming. International Journal of Advanced Manufacturing Technology, pp 1–12
7. El-Salhi S, Coenen F, Dixon C, Khan MS (2012) Identification of correlations between 3d surfaces using data mining techniques: predicting springback in sheet metal forming. In: International conference on innovative techniques and applications of artificial intelligence. Springer, pp 391–404
8. Khan MS, Coenen F, Dixon C, El-Salhi S (2012) Classification based 3-d surface analysis: predicting springback in sheet metal forming. J Theor Appl Comput Sci 6(2):45–59
9. Pinto T, Kohler C, Albertazzi A (2012) Regular mesh measurement of large free form surfaces using stereo vision and fringe projection. Opt Lasers Eng 50(7):910–916
10. Lu G, Sajjanhar A (1999) Region-based shape representation and similarity measure suitable for content-based image retrieval. Multimed Syst 7(2):165–174
11. Gill PE, Murray WMH (2021) Wright, numerical linear algebra and optimization. SIAM
12. Salomon D (2007) Curves and surfaces for computer graphics. Springer Science & Business Media
13. Egerton, PA, Hall W (1998) Computer graphics: mathematical first steps. Simon & Schuster International
14. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780
15. Russell S, Norvig P (2002) Artificial intelligence: a modern approach
16. Cortes C, Vapnik V (1995) Support-vector networks. Mach Learn 20:273–297
17. Graves A, Liwicki M, Fernández S, Bertolami R, Bunke H, Schmidhuber J (2008) A novel connectionist system for unconstrained handwriting recognition. IEEE Trans Pattern Anal Mach Intell 31(5):855–868
18. Graves A, Mohamed A-r, Hinton G (2013) Speech recognition with deep recurrent neural networks. In: 2013 IEEE international conference on acoustics, speech and signal processing. Ieee, pp 6645–6649
19. Theis L, Bethge M (2015) Generative image modeling using spatial lstms. Advances in neural information processing systems 28
20. Nair V, Hinton GE (2010) Rectified linear units improve restricted Boltzmann machines. In: Icml
21. Ortiz M, Penalva M, Iriondo E, López de Lacalle LN (2019) Accuracy and surface quality improvements in the manufacturing of ti-6al-4v parts using hot single point incremental forming. Metals 9(6):697
22. Scholkopf B, Sung K-K, Burges CJ, Girosi F, Niyogi P, Poggio T, Vapnik V (1997) Comparing support vector machines with Gaussian kernels to radial basis function classifiers. IEEE Trans Signal Process 45(11):2758–2765

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