Key Engineering Materials
ISSN: 1662-9795, Vol. 1050, pp 171-182
doi:10.4028/p-ls5PTZ

Submitted: 2026-01-19
Revised: 2026-02-16
Accepted: 2026-02-19
Online: 2026-04-16

Physics-Informed Recurrent Neural Networks with Kinematic Constraints for Large Deformation Metal Forming

Francesco Munzone[1,a*], Javad Hazrati[1,b], Wouter Hakvoort[2,c], Redmer Van Tijum[3,d] and Ton van den Boogaard[1,e]

[1]Nonlinear solid mechanics, Faculty of Engineering Technology, University of Twente, Enschede, Netherlands

[2]Control and Mechatronics, Faculty of Engineering Technology, University of Twente, Enschede, Netherlands

[3]Advanced Production Engineering, Engineering and Technology Institute Groningen, Faculty of Science and Engineering, University of Groningen, Netherlands

[a*]f.munzone@utwente.nl, [b]j.hazratimarangalou@utwente.nl, [c]w.b.j.hakvoort@utwente.nl, [d]redmer.van.tijum@philips.com, [e]a.h.vandenboogaard@utwente.nl

Keywords: physics-informed, recurrent neural network, deep-drawing, design exploration.

Abstract. Metal-forming is a manufacturing process that involves non-linear elastoplastic deformations to shape a blank into a complex geometry. These processes are governed by numerous parameters, with significant influence on the final product. To analyse the effects of such parameters, large-scale finite element (FE) simulations are often conducted. However, these models are computationally expensive and often unsuitable for real-time analysis. To overcome these limitations, surrogate models have emerged as powerful alternatives. In this study, we propose a physics-informed recurrent neural network framework (PI-RNN) to evaluate displacements and strain tensor components. In particular, the latter are not a network output but are obtained through the application of kinematic relations. Given the initial configuration as input and the final configuration as output, it is possible to evaluate the deformation gradient. The local condition of impenetrability of matter is then injected into the loss to improve the estimation of displacements and strain tensors. The PI-RNN model is trained on data generated from FE simulations of a deep-drawing process. The accuracy of the model is evaluated on a test dataset (design points that the model does not see during training) using different error measures. The results show that the proposed PI-RNN model can reproduce the FE simulation results fairly well.

Introduction

Most manufacturing processes aim to transform raw material into a complex-shaped final product via deformation steps. In sheet metal forming processes, the material is subjected to elasto-plastic deformations, in which many process parameters influence the quality of the final product. In manufacturing processes, design exploration, sensitivity analysis, and robust optimization are tasks that require thousands of simulations. Finite element (FE) softwares are typically employed to simulate the outcome of a forming process under different process conditions. This approach is generally considered expensive due to the number of simulations needed. A solution is offered by surrogate models, which act as approximators by learning the input-output relationship using data collected from expensive simulations. Once properly trained, they can replace the expensive model to provide fast and accurate predictions for previously unseen inputs. A widely employed class of surrogates is neural networks, which excel in learning complex and non-linear patterns, capable of handling (and preferred) in the case of large data. As for any surrogate models, neural networks learn the underlying system function based solely on data. The extraction of insight from the learned function is difficult due to the large number of parameters. Physics-informed neural networks (PINNs) offer a partial solution which relies on the automatic differentiation algorithm to compute partial derivatives of output with respect to inputs. This additional information is then injected into the loss of the network to force the solution to comply with the system equations. PINNs are widely

applied and studied in many scientific fields because of their ability to improve the generalization ability of the model based on knowledge of the system. Predictions are not purely data-driven, but also follow a logic dictated by the governing equations of the system, which usually takes the form of partial differential equations.

Elasto-plastic deformation problems are governed by complex mechanical and material equations. PINNs can exploit them to let the model find a solution that satisfies both data and governing laws. In the work of Niu et al. [1], a PINN framework is proposed to model elasto-plasticity in the case of large deformations. The model uses the initial configuration to predict displacements and the first Piola-Kirchhoff stress over the increment of a multi-step loading-unloading history. The loss includes equilibrium equations, boundary conditions, and constitutive relations, using automatic differentiation as a tool for the computation of partial derivatives. Another work by Maia et al. [2], in a multi-scale approach, substitutes the RVE micromodel of a unidirectional composite material (both path and time dependent behaviour) in the macro model using a physically recurrent neural network to describe the homogenized behaviour of the RVE in finite strain. In a later work [3], the model has been extended to consider anisotropic materials under the assumption of finite-strain.

Inspired by the current literature, we propose a physics-informed recurrent neural network (PI-RNN) to learn the deformation evolution of a metal sheet subject to deep-drawing. The model is trained to learn the deformation history under different process conditions using the initial configuration as input. This allows the network to compute the deformation gradient $\mathbf{F}$ through the evaluation of partial derivatives. The strain tensor components are then obtained by applying kinematic relations for large deformation. Moreover, $\mathbf{F}$ is used to enforce the local condition of impenetrability of matter on the loss function as a physical constraint [4]. The surrogate model is trained to learn the effect of different combinations of process parameters. To keep this initial study manageable, only the variations of the coefficient of friction and the thickness of the sheet are considered. An evolutionary stochastic enhanced Latin hypercube is used to generate a space-filling design of experiments (DOE).

The content of the article is organized as follows. The first section introduces the DOE algorithm used for the construction of the design space and the kinematic relations necessary to construct the loss function of the proposed physics-informed recurrent neural network. In the second section, the accuracy of the model and its generalization ability are assessed on a test dataset composed of unseen combinations of features. Finally, in the third section, the conclusive remarks and potential future work are discussed.

Physics-Informed Recurrent Neural Network

In this section, the surrogate modelling framework is discussed. Samples are selected from the design space using a search algorithm to find an optimal DOE, that is, a DOE in which the design points are spread out as evenly as possible across the design space. A physics-informed recurrent neural network is trained on the collected data. Partial derivatives of output variables with respect to inputs are exploited to evaluate the strain tensor via kinematic relations for large deformation.

Enhanced Stochastic Evolutionary Latin Hypercube.

In surrogate modelling, a proper design of experiments (DOE) is essential to achieve high accuracy over the design space. A typical procedure for the identification of an optimal DOE starts with a random design $\mathbf{X}_0$. An update algorithm is in charge of manipulating it and returning a new design. Then, the two designs are ranked based on a criterion, and it is decided whether to discard the new design or substitute the old design with the new. With Latin hypercube (LH), new designs can be easily generated through the exchange of columns or rows of the structure without altering the property of the LH.

The chosen criterion quantifies the space-filling quality of the current LH design. However, ranking every possible LH is impractical, especially for high-dimensional cases. In the literature, it is possible to find search algorithms whose job is to optimize an initial LH design, increasing its space-filling quality [5, 6]. In particular, the ESE algorithm proposed by Jin et al. (2005) [7], a variation of

the stochastic evolutionary algorithm proposed by Saab and Rao [8], uses a temperature parameter to balance global exploration and local exploitation.

The ESE algorithm consists of an inner loop and an outer loop. The inner loop is in charge of generating and accepting/rejecting new designs, based on the current threshold "temperature" T_h value. The current design $\mathbf{X}$ is randomly altered by swapping J columns, generating the design $\mathbf{X}_{try}$. The values of the optimal criterion allow the algorithm to accept or reject the $\mathbf{X}_{try}$, based on the T_h. A function that generates a uniform random number between 0 and 1 is multiplied by T_h. This allows the algorithm to choose a better or a slightly worse designs, promoting exploration of the design space. The outer loop controls the optimization process by adjusting T_h. If a good design is found, T_h is adjusted to locally search for the new optimal design. If no improvement is detected, T_h is adjusted such that the algorithm searches for a combination of design points, escaping local minima. The parameters α_1, α_2, and α_3 in charge of controlling T_h are kept as given in the original work [7].

The optimal criterion chosen is the maximin criterion proposed by [9]. The optimal design is the one that minimizes the following function:

$$\phi_p(X) = \left[\sum_{j=1}^{m} J_j d_j^{-p}\right]^{-p} \tag{1}$$

where d is a list $(d_1, d_2, \ldots, d_m)$ of distances in which d_j represents the distance between two sample points in the design, and J is an index list $(J_1, J_2, \ldots, J_m)$ in which J_j is the number of pairs separated by the distance d_j. The exponent is a positive integer. Here, a value of $p = 50$ is chosen.

Deep drawing forming process.

The case study is a single deep drawing stage, shown in Fig. 1. This forming process has been designed by Veldhuis et al. [10] to be sensitive to temperature, such that friction variations are caused by temperature-induced effects. Indeed, their sensitivity analysis showed that the coefficient of friction (COF) is one of the most dominant factors for the final critical-to-quality parameters (CTQs) such as the cup height, inner diameter, and outer diameter. Another parameter that showed to have a mild influence on the CTQs is the sheet thickness (th), which may vary within the metal sheet roll. Based on this, the design space is built considering the two aforementioned features.

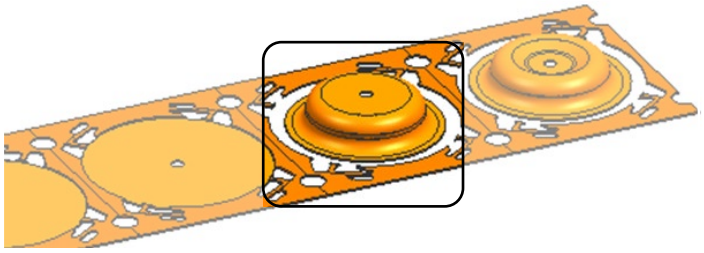

Fig. 1. The metal forming process used as a case study is the first stage of a deep drawing process.

Based on the work of Jin et al. (2001) [11], a total of 40 design points (that is, FE simulations) are chosen to train the model. The test dataset is built using the same ESE-LH algorithm by sampling design points outside the training dataset. The respective range for each feature is based on the intervals reported in [10], which are $[0.05, 0.2]$ for COF and $[0.29, 0.31]$ for th. The FE software used is Marc Mentat 2022.1. The sheet is modelled as an axisymmetric object, with 300 elements over radial direction and 6 elements over thickness direction, for a total of 1800 quadrilateral linear elements.

Kinematic relations for large deformations.

In sheet metal forming and forming processes in general, the workpiece is subject to large elasto-plastic deformation. Infinitesimal strain theory does not hold up, and it is necessary to switch to a

more advanced theoretical formulation. Proper evaluation of strain measurement requires considering large deformation and large rotations.

The following derivations are based on Simo and Hughes work [4]. Let $\mathcal{B}$ be the reference configuration of a body with particles $\mathbf{X} \in \mathcal{B}$. With φ we describe the displacements of this body to the current configuration $\mathcal{S}$:

$$\varphi: \mathcal{B} \subset \mathbb{R}^3 \rightarrow \mathcal{S} \subset \mathbb{R}^3 \tag{2}$$

The function φ maps every particle $\mathbf{X} \in \boldsymbol{\mathcal{B}}$ to $\mathbf{x} \in \mathcal{S}$. We can define the deformation gradient as the derivative of the deformation:

$$\mathbf{F}(\mathbf{X}) = \frac{\partial \varphi(\mathbf{X})}{\partial \mathbf{X}} \tag{3}$$

The deformation gradient allows us to define the unphysical condition for which the volume of the particle cannot be smaller than zero:

$$\mathrm{J}(\mathbf{X}) = \det \mathbf{F} > 0. \tag{4}$$

Through the deformation gradient, the left Cauchy-Green tensors are defined:

$$\mathbf{B} = \mathbf{F}\mathbf{F}^{\mathrm{T}} \tag{5}$$

The deformation gradient is an invertible second-order tensor; thus, it can be decomposed through the polar decomposition theorem:

$$\mathbf{F} = \mathbf{R}\mathbf{U} = \mathbf{V}\mathbf{R} \tag{6}$$

where $\mathbf{R}$ is a proper orthogonal tensor, and $\mathbf{U}(\mathbf{X})$, $\mathbf{V}(\mathbf{x})$ are symmetric, positive-defined tensors called the right and left Cauchy-Green tensors. In particular, Eq. 5 can be rewritten as follows:

$$\mathbf{B} = \mathbf{V}^2 \tag{7}$$

The tensor $\mathbf{B}$ is symmetric and positive definite, hence by the spectral theorem:

$$\mathbf{B} = \sum_{\mathrm{i}}^{3} \lambda_{\mathrm{i}}^2 \mathbf{n}^{(\mathrm{i})} \otimes \mathbf{n}^{(\mathrm{i})} \tag{8}$$

Consequently, the left stretch tensor can also be written in terms of eigenvalues and eigenvectors:

$$\mathbf{V} = \sum_{\mathrm{i}}^{3} \lambda_{\mathrm{i}} \mathbf{n}^{(\mathrm{i})} \otimes \mathbf{n}^{(\mathrm{i})} \tag{9}$$

A widely used strain measure is the logarithmic strain, or Hencky strain. From Eq. 9, the Hencky strain can be written as:

$$\ln \mathbf{V} = \sum_{\mathrm{i}}^{3} \ln \lambda_{\mathrm{i}} \, \mathbf{n}^{(\mathrm{i})} \otimes \mathbf{n}^{(\mathrm{i})} \tag{10}$$

Physics-Informed Recurrent Neural Networks.

Physics-informed neural networks (PINNs) are a class of neural networks in which the governing equations (or physical relations) of the system are placed into the model loss function as additional penalty terms. Generally, the physical laws describing a system are in the form of partial differential equations or involve partial derivatives. Moreover, if known, boundary conditions and initial conditions can also be as additional constraint in the loss function due to their involvement in solving PDEs. To evaluate partial derivatives, PINNs take advantage of the automatic differentiation (AD)

algorithm. Nevertheless, it is not prohibited to use numerical methods for the evaluation of such derivatives. The general form of the loss function is the following:

$$\mathcal{L} = \mathcal{L}_{\text{data}} + \mathcal{L}_{\text{PDE}} + \mathcal{L}_{\text{BC}} + \mathcal{L}_{\text{IC}} \tag{11}$$

where each term is referred to as labelled data ($\mathcal{L}_{\text{data}}$), PDEs solution ($\mathcal{L}_{\text{PDE}}$), boundary conditions ($\mathcal{L}_{\text{BC}}$), and initial conditions ($\mathcal{L}_{\text{IC}}$).

Many variants of PINNs exist based on the type of architecture used. Here, the problem under study is a sheet metal forming problem, thus subject to plasticity and large deformation. The history of the process plays a crucial role in the evolution of the geometry. Therefore, recurrent neural networks are a reasonable choice as the base architecture for the network. The model takes as inputs the initial configuration, the displacement of the punch, the time vector, the coefficient of friction, and the sheet thickness. The initial configuration spatial coordinates, the displacement, and the strain components are sampled at the boundary nodes and at random integration points. The coefficient of friction and the thickness are used, instead, to build the DOE. The primary output of the network are the displacements $\mathbf{x}$ throughout the deformation process, whereas the components of $\ln \mathbf{V}$ are secondary quantities computed from it via the kinematic relations. The relation between the initial configuration (inputs) and the current configuration over time (output) allows us to evaluate $\mathbf{F}$ through the AD algorithm, and the kinematic relations to evaluate the components of $\ln \mathbf{V}$.

The current PI-RNN model, does not utilize the classic loss terms introduced in Eq. 11; that is, the PI-RNN does not leverage partial derivatives to solve a PDE. Both displacement and strain loss terms are evaluated through the mean square error using the data collected from FE simulations. The physical constraint is represented by the local condition of impenetrability of matter introduced in Eq. 4. Therefore, the loss function can be written as follows:

$$\mathcal{L} = \mathcal{L}_{\text{disp}} + \mathcal{L}_{\text{strain}} + \mathcal{L}_{\text{imp}} \tag{12}$$

where each term is described as follows:

$$\mathcal{L}_{\text{disp}} = \|\mathrm{x_i} - \mathrm{x_i^*}\|^2, \tag{13}$$

$$\mathcal{L}_{\text{strain}} = \left\|(\ln \mathrm{V})_{\mathrm{ij}} - (\ln \mathrm{V})_{\mathrm{ij}}^*\right\|^2, \tag{14}$$

$$\mathcal{L}_{\text{imp}} = \ln\left(1 + \mathrm{e}^{-\mathrm{k}\det \mathbf{F}}\right). \tag{15}$$

The factor k in Eq. 15 is necessary to reduce the slope of the softplus function and make the contribution of this loss term negligible whenever the local impenetrability condition uphold (Eq. 4). In particular, for metal deformation problems $\det \mathbf{F}$ is close to one. The values for k is manually tuned beforehand to ensure a that the transition between negative and positive values of $\det \mathbf{F}$ is sharp, but still smooth. Here, k is set to 30.

It is worth noting that no scaling weights are employed in the current implementation. Instead, differences in scale are addressed by standardizing both inputs and outputs. This ensures that the loss terms remain comparable in magnitude, preventing any single term from dominating the optimization process.

Learning rate tuning.

Due to the contribution of partial derivatives in the loss function during training, PINNs becomes sensitive to the choice of learning rate. An efficient technique to assess an optimal learning rate value is to train the model for a small number of epochs, with the learning rate increasing over the iterations. As the learning rate increase, the model learns faster, resulting in lower values of the loss function, until the learning rate becomes too large, causing the model to fail.

The learning rate is increased from 10^{-10} to 0.1 using the following evolution formula:

$$\beta = \beta_0 \left(\frac{\beta_f}{\beta_0}\right)^{\gamma} \tag{16}$$

with β_0 the initial learning rate, β_f the final learning rate, and with γ the ratio between the current number of iterations and the maximum number of iterations. The number of epochs, thus the number of iterations, is arbitrary. A good trade-off is a maximum number of iterations that ensure a smooth evolution of β and a rapid evaluation of the loss curve in prediction. Here, it is chosen to train the model for approximately 1000 iterations, corresponding to 10 epochs. The algorithm is run on an architecture consisting of 2 hidden layers and 128 nodes. In Fig. 2, the validation loss evolution versus the learning rate is shown. Smith [12] suggests two methods to determine the optimal learning rate.

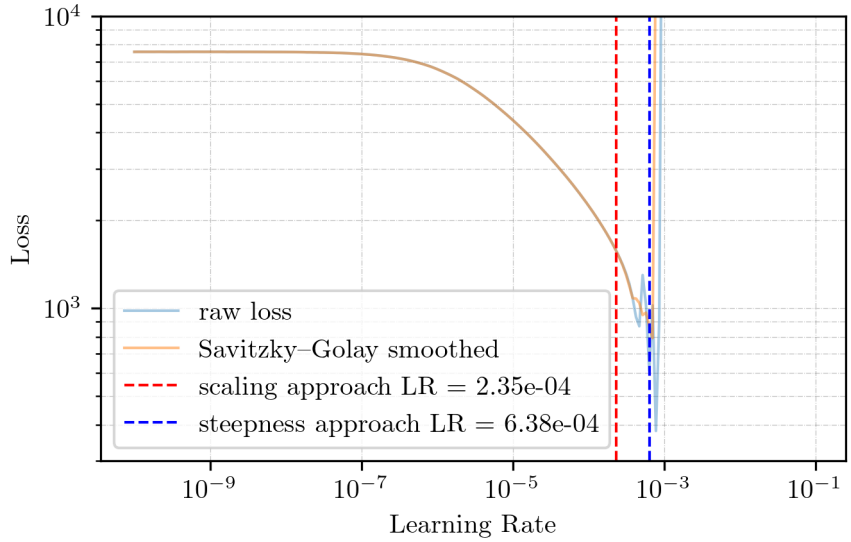

Fig. 2. Learning rate versus validation loss used to identify the optimal learning rate range.

The first method relies on the minimum of the curve, corresponding to the point after which the model starts to diverge and explode. In fact, in the article is suggested to set $\beta = \beta_{min}/3$. With this formula we find an optimal $\beta = 2.35 \cdot 10^{-4}$. The second method, which the same author points out as the most reliable, is to select the point at which the validation loss is the steepest. However, it can be observed that the light blue curve in Fig. 2 is slightly noisy, which makes the identification of the region of interest difficult. Hence, the validation loss curve is smoothed using a Savitzky-Golay filter. The resulting smooth curve allows us to identify the point at which the loss is steepest, corresponding to a $\beta = 6.35 \cdot 10^{-4}$. For most trainings performed in this article, the latter value is selected as the default learning rate enabling faster learning. However, its vicinity to the instability region may lead to model failure. Thus, the learning rate is kept flexible and open to being varied in the region defined by the two optimal values if difficulties in training are encountered.

Results

Impact of model architecture on accuracy.

The architecture of a neural network dictates the building blocks of the model, necessary to construct the input-output nonlinear relationship. In particular, the number of trainable model parameters is commonly used to define the complexity of the model. It is common knowledge that complex models are more versatile due to the high number of parameters but prone to overfitting. On the other hand, less complex models are less troublesome to train, but may not fully learn the underlying system relationship, resulting in underfitting.

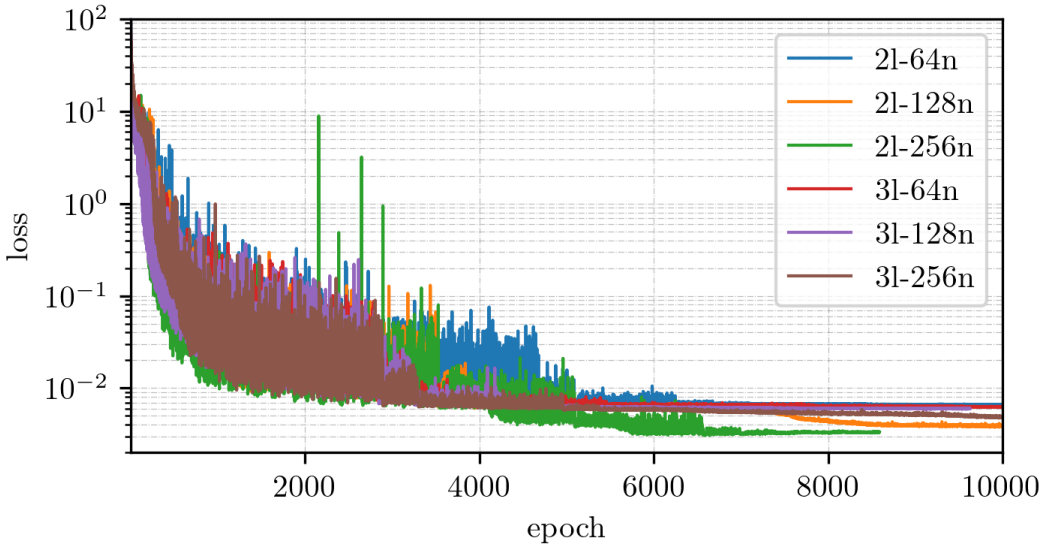

Fig. 3. Validation loss versus epochs for different architectures.

Different architectures are evaluated to check for complexity limits of the model. The architecture of the PI-RNN involves a set of stacked gated recurrent units (GRUs) as hidden layers and a linear layer as the output layer. The activation functions for the gates and for the recurrent unit are, respectively, the sigmoid and the hyperbolic tangent function [13]. The combination of layers and nodes is decided based on a full factorial design, in which the number of layers is $l = \{2, 3\}$ and the number of nodes (per layer) is $n = \{64, 128, 256\}$. The objective is not to find the optimal architecture, since this requires a much deeper analysis, but to assess the sensitivity of the model to these hyperparameters and check if model complexity leads to overfitting or underfitting. The models are implemented in the Tensorflow (v2.19) framework, trained using GPUs (NVIDIA L4). The optimizer used is Nadam with a learning rate chosen as defined in the previous section.

In Fig. 3, we can observe the validation losses over the epochs for the different architectures. The best validation loss is obtained by the model using 2 layers and 256 nodes, followed by the model with 2 layers and 128 nodes. All the remaining models seem to converge towards the same solution. From the validation losses, it can be noticed that the model responds better when the number of nodes is increased rather than the number of layers. In fact, it is typical for deeper networks to be more difficult to train, leading to accuracy degradation [14]. For the purpose of assessing model complexity, it is not necessary to tune the learning rate for each architecture, as long as the training converges towards an optimal solution. To avoid overfitting, in training, the learning rate is halved whenever the validation loss hits a plateau for more than 400 epochs (observe the drops in the losses in Fig. 3). A single GPU is able to host between 2 and 3 models before running out-of-memory. This influences computational time. Generally, a training time per epoch is observed between 45 and 50 seconds.

The learning rate is set to $6.35 \cdot 10^{-4}$ for all models, except for the models using 2 layers and 128 nodes, 3 layers and 64 nodes, and 3 layers and 256 nodes, where a learning rate of $4 \cdot 10^{-4}$ is used due to multiple failure of the model caused by the explosion of the loss term $\mathcal{L}_{imp}$. The unstable behaviour of $\mathcal{L}_{imp}$ is, however, still observable in Fig. 4 even after the adjustment of the learning rate. The cause seems not to be related to the model architecture, but rather on the random initialization of the model parameters. A solution may be given by initially warming up the model using lower learning rate values (chosen in the optimal region) till convergence is found. Then the learning rate is let increase over training to prevent the model to get stuck in (or unable to escape) local minima.

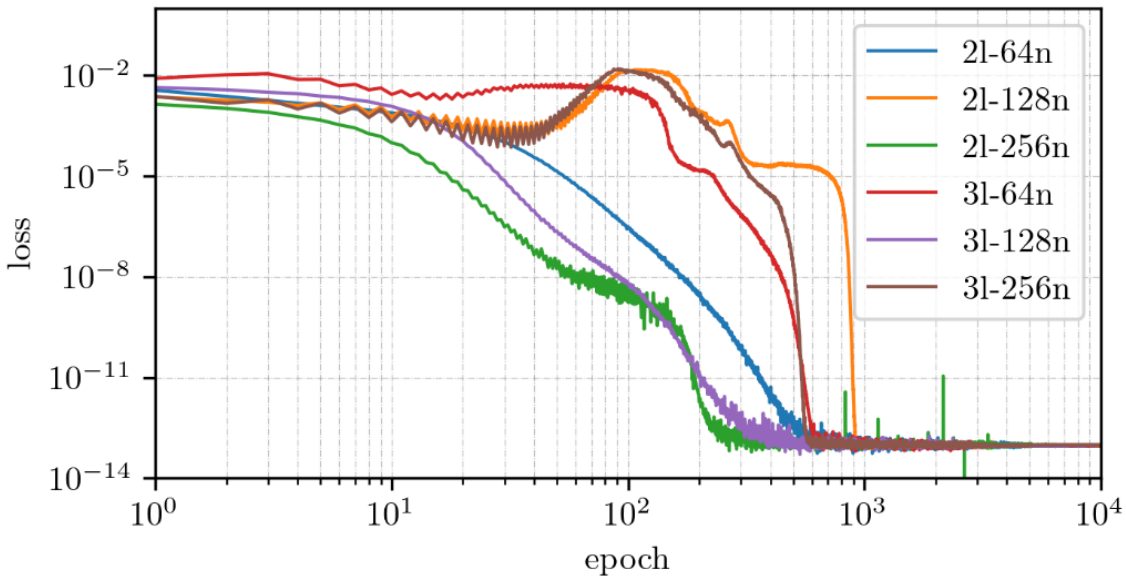

Fig. 4. Validation $\mathcal{L}_{imp}$ versus epochs for different architectures.

Predictions on a test dataset.

The test dataset is a composition of data collected from FE simulation performed using feature combination not seen by the model during training. The ESE algorithm is again employed to generate a test DOE. It is ensured that the resulting design points are distinct from those in the training DOE. The trained model evaluates the predictions at nodal spatial coordinates. With the only exception of the nodes at the boundaries, the nodal coordinates are unseen data points for the model. Moreover, the boundary values are still coming from a combination of friction and thickness never been observed by the model.

To assess the accuracy of the model over the design space, the fraction of variance unexplained (FVU) is used, which can be defined in terms of the coefficient of determination:

$$FVU = \frac{SS_{res}}{SS_{tot}} = \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2} = 1 - R^2. \tag{17}$$

The nominator SS_{res} is the sum of squares of residuals, in which y_i is the true value and $\hat{y}_i$ is the predicted values. The denominator SS_{tot} is the total sum of squares, in which $\bar{y}$ is the mean over all observations. In terms of R^2, a surrogate model that returns a value of approximately 1, means that SS_{res} is approximately zero, and the surrogate is an optimal representation of the system. On the other hand, an R^2 close to zero means that the surrogate has the same representation ability as a model that always predicts $\bar{y}$.

It is interesting to observe the FVU strain components over the DOE in Fig. 5. Here, the mean value $\bar{y}$ is evaluated as the mean value per design point, that is, the mean evaluated using the collection of observations in space and time of one simulation. The FVU values are small overall, indicating the good approximation provided by the PI-RNN. However, the strain components $ln\,V_{12}$ and $ln\,V_{22}$ show a higher FVU for the combination of feature $COF = 0.1962$ and $th = 0.3085$ (simulation 17). Investigations on simulation 17 showed a concentration of error for both $ln\,V_{12}$ and $ln\,V_{22}$ in the spatial region in which the sheet comes in contact with the punch. For simplicity, only the absolute error $AE_{ln\,V_{12}}$ is analysed. Every conclusion can also be applied to $AE_{ln\,V_{22}}$. The error appears in the early stages of the deformation process (Fig. 6 (right)), but no irregularities are spotted in the FE results (Fig. 6 (left)).

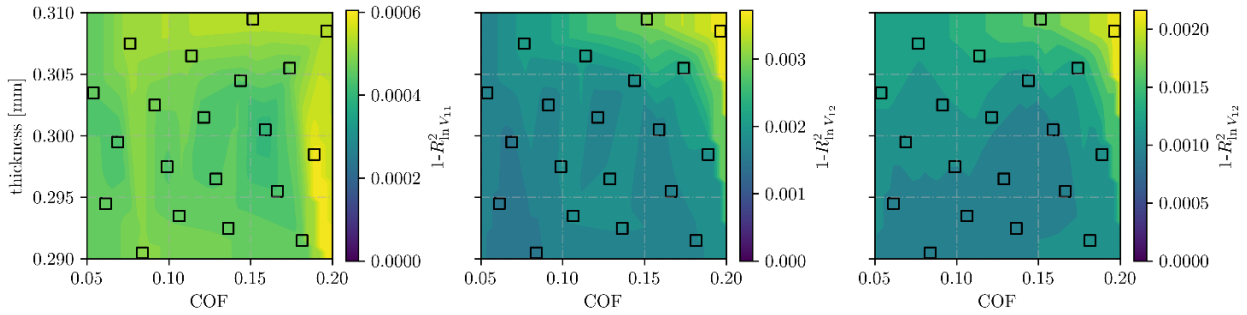

Fig. 5. FVU over the design space for the strain $\ln V_{11}$ (left), $\ln V_{12}$ (center), and $\ln V_{22}$ (right). The black squares represent the test design points.

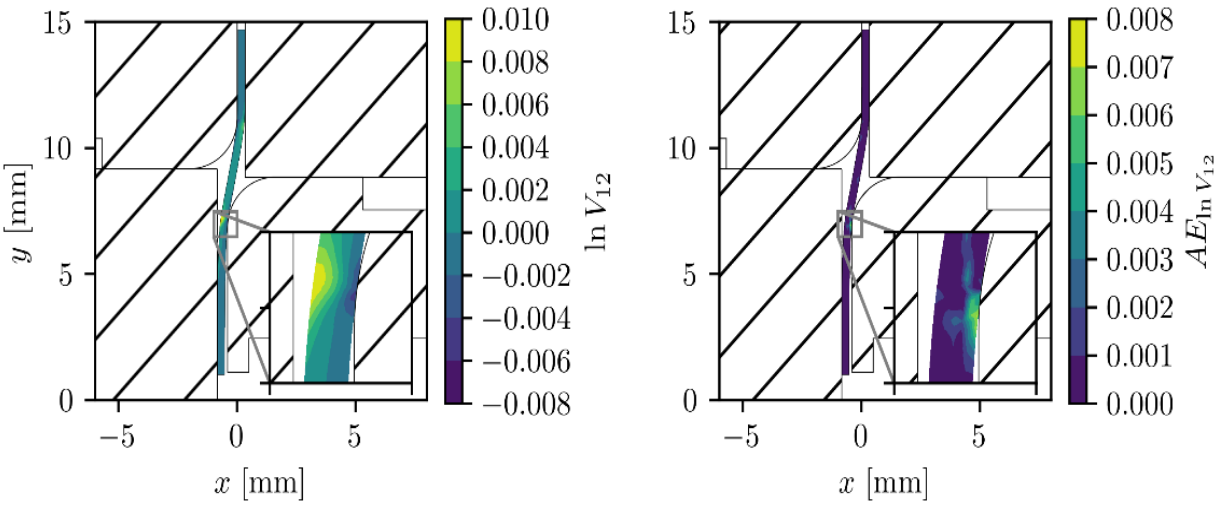

Fig. 6. FE prediction (left) and absolute error (right) for $\ln V_{12}$ at increment 167 of simulation 17, part of the test dataset.

Thus, the cause of such behaviour must fall on the PI-RNN and, in particular, on the training dataset. With this in mind, the FVU of each quantity is evaluated again over the design space using the training dataset. In Fig. 7 we can observe that for certain combinations of COF and th, all the fields show the same peaks of error. In particular, we will refer to simulation 16 as the combination of $\mathrm{COF} = 0.1568$ and $\mathrm{th} = 0.2937$ mm, which corresponds to the simulation returning the highest FVU values for $\ln V_{11}$.

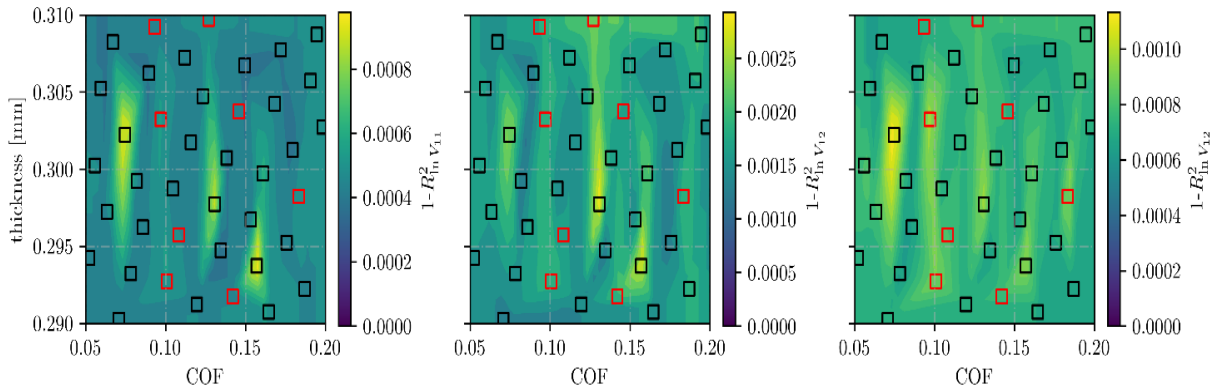

Fig. 7. FVU over the design space for the strain $\ln V_{11}$ (left), $\ln V_{12}$ (center), and $\ln V_{22}$ (right). The black squares represent the training design points, while the red the validation design points. The validation set is created by randomly splitting the original dataset into training and validation dataset with an 80:20 ratio.

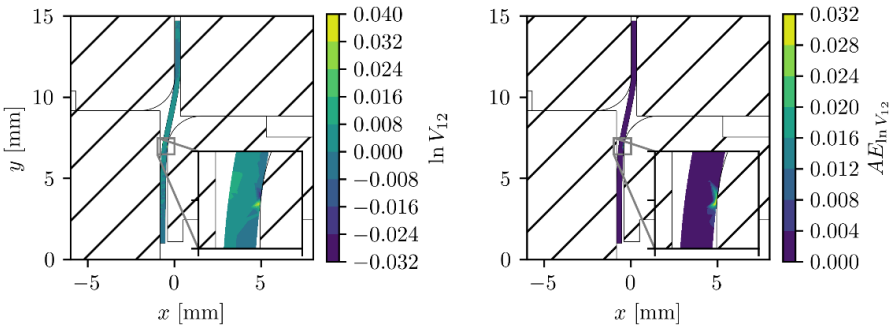

Fig. 8. FE prediction (left) and absolute error (right) for $\ln V_{12}$ at increment 167 of simulation 16, part of the training dataset.

Again, in the early stages of the deformation process, the error is located in the same region as the one observed in Fig. 6, but much higher in comparison. In particular, we can notice that in the FE simulation results, there is a jump in correspondence with the error concentration (Fig. 8 (left)), which is caused by a mesh distortion. Thus, the PI-RNN is learning the input-output relation based on unphysical conditions, leading to wrong predictions. In fact, the same mesh distortion is not observed in simulation 17, but the PI-RNN still predicts a jump in the strain components that does not exist. This can be avoided by examining the training dataset carefully, possible when dealing with a small dataset, but impractical for large datasets due to the mole of data that should be analysed. Nevertheless, more attention should be paid to the preprocessing of training data to discard possible poor FE simulations.

Fig. 9 and Fig. 10 show predictions versus true plots. The narrow blue band in Fig. 10 represents the 95% prediction interval; thus, the points falling outside the band are the remaining 5%. As observed in the FVU, the strain components show higher inaccuracy compared to the displacements. Displacements are a direct output of the PI-RNN, therefore easy to fit, while the strain comes from the various transformations of the deformation gradient, making fitting more complex, but still sufficiently accurate.

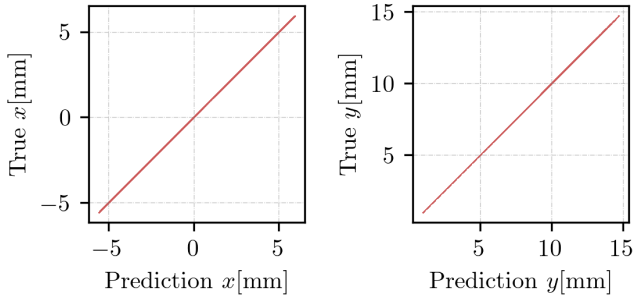

Fig. 9. Prediction versus true of the displacements.

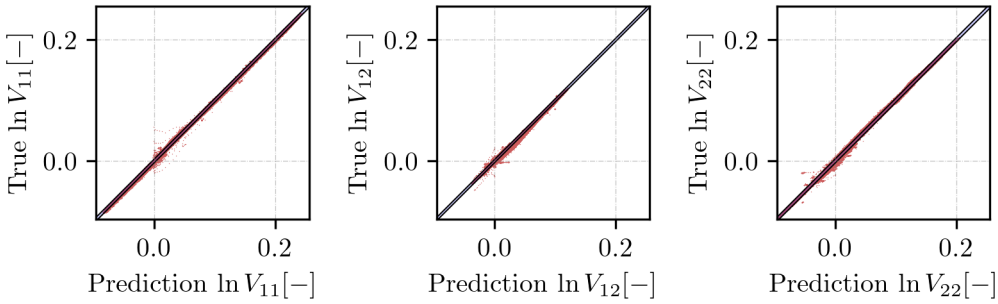

Fig. 10. Prediction versus true of the strain components. The blue band represents the 95% prediction interval.

Conclusion

In this work, a recurrent neural network is used to learn the deformation evolution of a deep drawing process under different process conditions. The logarithmic strain tensor is evaluated using the deformation gradient obtained as the partial derivatives between the initial and current configuration, respectively input and output of the model. Moreover, **F** is used to impose locally the impenetrability of matter on the loss function as an additional constraint.

The PI-RNN is trained on data collected from FE simulations of the deep drawing process. Each simulation corresponds to a design point sampled in the design space with the help of the ESE-LH algorithm. The model is trained on displacement and strain components values sampled at the boundary nodes and at random integration points.

A preliminary analysis is performed to choose the optimal learning rate value to avoid model failure. Based on the literature, an optimal range for learning rate is found $2.35 \cdot 10^{-4} \leq \beta \leq 6.35 \cdot 10^{-4}$. Moreover, different architectures are tested using different combinations of layers and nodes to observe the complexity limitation of the model. All architectures converge towards a similar suboptimal minimum, with the best performance achieved by the model using 2 layers and 256 nodes. This investigation also showed the inclination of the model to fail due to instabilities of the loss term $\mathcal{L}_{\text{imp}}$, unrelated to the model architecture. The cause can be found in the random initialization of the model parameters.

The generalization ability of the model is tested using 20 design points unseen by the model during training. The FVU over the design space showed that the model is perfectly capable of predicting both displacement and strain components with high precision. The output standardization works well in putting the scalar quantities on a comparable scale. However, the PI-RNN generalization ability could be further improved by introducing weighting factors for each loss term to balance their contribution during the optimization process. Adaptive weighting methods based on gradient information may prove necessary to escape eventual local minima during training. The latter will be a subject of future work.

The current application of the PI-RNN considers a design space consisting of two features. In metal forming processes, many parameters play a crucial role in the shape of the final geometry. A foreseen extension of the model is the inclusion of additional features to perform sensitivity analysis and design exploration in the case of a larger DOE.

Acknowledgment

This publication is part of the project Digital Twin (project 2.2) with project number P18-03 of the research programme Perspectief, which is (mainly) financed by the Dutch Research Council (NWO).

References

[1] S. Niu, E. Zhang, Y. Bazilevs and V. Srivastava, “Modeling finite-strain plasticity using physics-informed neural network and assessment of the network performance”, J. Mech. Phys. Solids, vol. 172, no. 0022-5096, p. 105177, 2023.

[2] M. Maia, I. Rocha, D. Kovačević and F. v. d. Meer, “Physically recurrent neural networks for path-dependent heterogeneous materials: Embedding constitutive models in a data-driven surrogate”, Comput. Methods Appl. Mech. Eng., vol. 407, no. 0045-7825, p. 115934, 2023.

[3] M. Maia, I. Rocha, D. Kovačević and F. v. d. Meer, “Physically recurrent neural network for rate and path-dependent heterogeneous materials in a finite strain framework”, Mech. Mater., vol. 198, no. 0167-6636, p. 105145, 2024.

[4] J. C. Simo and T. J. R. Hughes, Computational Inelasticity, New York: Springer, 1998.

[5] Li, W. W. and C. F. Jeff Wu, “Columnwise-Pairwise Algorithms with Applications to the Construction of Supersaturated Designs.,” Technometrics, vol. 2, p. 171–179, 1997.

[6] K. Q. Ye, W. Li and A. Sudjianto, “Algorithmic construction of optimal symmetric Latin hypercube designs”, J. Stat. Plan. Inference , vol. 90, no. 0378-3758, pp. 145-159, 2000.

[7] R. Jin, W. Chen and A. Sudjianto, “An efficient algorithm for constructing optimal design of computer experiments”, J. Stat. Plan. Inference, vol. 134, no. 1, pp. 268-287, 2005.

[8] Y. G. Saab and V. B. Rao, “Combinatorial optimization by stochastic evolution”, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol. 10, no. 4, pp. 525-535, 1991.

[9] M. D. Morris and T. J. Mitchell, “Exploratory designs for computational experiments”, J. Stat. Plan. Inference, vol. 43, no. 3, pp. 381-402, 1995.

[10] M. Veldhuis, J. Heingärtner, A. Krairi, D. Waanders and J. Hazrati, “An Industrial-Scale Cold Forming Process Highly Sensitive to Temperature Induced Frictional Start-up Effects to Validate a Physical Based Friction Model”, Procedia Manuf., vol. 47, no. 2351-9789, pp. 578-585, 2020.

[11] R. Jin, W. Chen and S. T. W., “Comparative studies of metamodelling techniques under multiple modelling criteria”, Struct. Multidiscip. Optim., vol. 23, no. 1, pp. 1-13, 2001.

[12] L. N. Smith, “Cyclical Learning Rates for Training Neural Networks”, in IEEE WACV, 2017.

[13] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk and Y. Bengio, “Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling”, arXiv, 2014.

[14] K. He, X. Zhang, S. Ren and J. Sun, “Deep Residual Learning for Image Recognition”, in IEEE CVPR, 2016.