



# Recurrent and convolutional neural networks in structural dynamics: a modified attention steered encoder–decoder architecture versus LSTM versus GRU versus TCN topologies to predict the response of shock wave-loaded plates

Saurabh Balkrishna Tandale<sup>1</sup> · Marcus Stoffel<sup>1</sup>

Received: 29 April 2022 / Accepted: 8 March 2023 / Published online: 31 March 2023  
© The Author(s) 2023

## Abstract

The aim of the present study is to analyse and predict the structural deformations occurring during shock tube experiments with a series of recurrent and temporal convolutional neural networks. The goal is to determine the architecture that can best learn and predict physically and geometrically nonlinear deformations. For this approach, four different architectures are proposed. Firstly, the multi-layered long-short term memory is developed followed by the multi-layered gated recurrent unit (GRU). Both the RNNs allow accounting for history dependent behaviors through their corresponding internal variables. Further, a multilayered temporal convolutional network is initialized, where the dilated convolution operation is responsible for tracing the path dependent behavior. In the mentioned architectures a sequence of mechanical data is passed through the network and a transformation to corresponding displacements is established. These sequences and corresponding deflections belong to a wide range of strain rates in the dynamic response of structures consisting of steel, aluminum, and copper plates including geometrical and physical non-linearities. Finally, an encoder–decoder architecture consisting of GRU layers is introduced with a modified attention mechanism which showed the best result for predicting the dynamic response. Employing comparative calculations between the neural network (NN) enhanced predictions and the measurements, the nature of approximation of each mentioned NN architecture is discussed and the capabilities of these developed surrogate models are demonstrated by its prediction on validation experiments. These validation experiments have displacement and input data ranges beyond the range of data used for training the aforementioned models.

**Keywords** Recurrent neural network · Shock tube experiments · Encoder–decoder · Attention · Temporal convolutional network · Viscoplasticity

## 1 Introduction

With recent breakthroughs in deep learning, artificial intelligence has promoted faster solutions of complex numerical tasks in several fields [1]. The successful employment of NNs in engineering applications as an alternative to continuum mechanics is well recorded in recent years. The trend was evident in [2] where the one-dimensional stress states of

the metal specimen were approximated using a feed-forward neural network (FFNN). The problems related to vibration and stability of the structures were discussed in [3,4]. In [5], a comparison between the stability of FFNN and a structural model for high-speed deformations is presented. Heider et al. [6] proposed an informed graph-based neural network to investigate frame invariance in anisotropic elastoplastic materials. A Monte Carlo strategy was also developed to evaluate the response statistics of nonlinear structural dynamics using FFNN and convolutional neural networks (CNNs) in [7]. Furthermore, due to the compatibility of the NN to simulate the nonlinear material behavior at the Gaussian points, the NNs were used as surrogate models in multiscale scenarios to learn history-dependent constitutive laws in [8] where an FFNN is used to replace the damage model proposed by

---

✉ Marcus Stoffel  
stoffel@iam.rwth-aachen.de  
Saurabh Balkrishna Tandale  
tandale@iam.rwth-aachen.de

<sup>1</sup> Institute of General Mechanics, RWTH Aachen University, Aachen, Germany

Chaboche [9]. Similarly, in [10,11] the viscoplastic structural response was learned by an FFNN, and a nonlinear effective electric constitutive law was developed in [12]. Theocaris and Panagiotopoulos proposed fully connected FFNN to model kinematic hardening [13] and identify the parameters for the failure mode of anisotropic materials [14]. A feed forward neural network however, lacks the ability to capture the loading history by itself and, thus, a modified recurrent neural network was proposed by Ghavamian and Simone in [15] to accelerate the 1-D Multiscale Finite Element simulations. Similarly in [16], multidimensional history-dependent multiscale simulations were successfully performed using GRU [17] with high-speed gains. Gorji et al. [18] also proposed a combination of GRU and fully connected layers to predict the 2D plane stress plasticity with anisotropic hardening for arbitrary loading paths. A physics based self-learning strategy was introduced in [19] to compute viscoplastic structural response. A data-driven paradigm was introduced in the studies [20–23] where the material behavior was learned directly from the experimental data skipping the part of empirical modeling. In [24], Capuano and Rimoli proposed smart elements with linear and non-linear elastic behavior approximating the element stiffness matrix using the FFNN. This work replaced the need for integration points as the element force vector was computed by the forward pass of the network and the stiffness was computed by perturbation technique. A further enhancement concerning the new Sobolev training strategy was introduced for stiffness replacement approach in [25,26]. More recently, model order reduction with sub-structuring was enhanced with so-called non-linear meta elements in [27], where the history-dependent response of the substructures or patches of elements are approximated through a TRU-Net architecture. Thus, with the context of neural networks in structural mechanics, five research directions can be identified namely constitutive modeling, surrogate modeling, stiffness modeling, multi-scale simulations, and sub-structuring. In the present study, surrogate modeling is chosen to predict efficiently the time-variant dynamic response of shock wave-loaded metal plates. In previous studies of the authors [28], these dynamic responses were captured using CNNs and a novel modular radial basis function network (RBFNs). However, in the current study, the advantage of gated RNNs to trace the path dependency with its internal variables and the Temporal Convolutional network with its dilated operation [29–31] is utilized as an alternate solution to passing time as an additional input as proposed in [28].

The objective of the present study is to propose an encoder–decoder architecture comprising of GRU [17] layers wherein the encoder is responsible for encoding the input mechanical data in a fixed-length vector. This encoded information is used by the decoder to generate the corresponding displacement sequences. A similar approach was adopted

in [32] to accurately compute the material response. To the best of the author's knowledge, the use of the mentioned architecture to predict displacement sequences in non-linear structural dynamics is new. The benefit of this approach will be to predict by neural networks sequences of structural deformation outside of the training range.

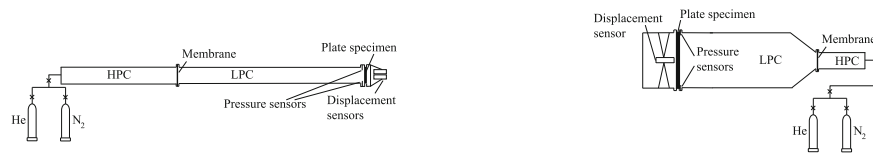
An architecture as used here can be analogously compared with the sequence to sequence machine translation task. This idea was originally presented in [33] for language translation and semantic sentence representations. In [34] the architecture was enhanced even further by the introduction of an attention mechanism. The current research addresses the computational bottleneck of fixed encoded representation for sequence problems also in dynamics and proposes a modified attention mechanism that allows the model to softly allocate probability values for parts of the encoded input sequence that are relevant for predicting the displacement at the next time step. This modified approach will be presented along with multilayered LSTM [35], GRU and TCN models for predicting the non-linear dynamic response. In comparison to the state of the literature, the present study proposes the following developments.

- Gated recurrent units and temporal convolution to predict history-dependent dynamic responses of shock wave loaded plate elements.
- An encoder–decoder architecture based on gated RNNs to predict the nonlinear structural behavior.
- A modified attention mechanism to dynamically compute the relevance of each encoded data point to predict the displacement at the following time step.
- This strategy allows to predict not only single data points but entire sequences of structural deformations outside of the trained data range.

The efficiency of the proposed attention steered encoder–decoder architecture will be demonstrated by using the experimental data mentioned in [28,36]. The ability of this proposed architecture is compared with multilayered LSTM, GRU and TCN networks especially for the validation experiments and the difference in the learning strategies of the aforementioned architectures are discussed.

The present research is organized as follows. In Sect. 2, the details of the experiments are highlighted followed by the NN formulations and in Sect. 3, the sampling procedure along with the gated recurrent and temporal convolutional neural networks are discussed. This builds up an intuition for an encoder–decoder architecture with gated RNNs followed by a modified attention mechanism. In Sect. 4, the data processing along with the optimization of hyperparameters is described. In Sect. 5, the results from multilayered LSTM, GRU, TCN networks, and the encoder–decoder architecture are presented. Further, the nature of the function approxi-

**Fig. 1** Experimental setups. **a** Experimental setup: plate specimen copper or steel. **b** Experimental setup: plate specimen aluminum.



(a) Experimental setup: Plate specimen copper or steel

(b) Experimental setup: Plate specimen aluminum.

mated by all the aforementioned networks is discussed along with their predictions for validation experiments. Finally, it is concluded that the proposed encoder–decoder architecture with attention had the best generalization capability and the results with a series of validation experiments are presented.

## 2 Experiments

In the current research, the shock tube experiments [37–40] are chosen covering a wide range of strain rates in the dynamic response of structures. It includes complex strain-rate dependent evolutions of geometrically and physically non-linear structural deformation. Pressure and deformation histories are provided to the aforementioned NN models. In these experiments, to account for different materials and diameters, two experimental setups are presented in Fig. 1a, b. Steel, Aluminum, and copper plates are subjected to impulsive loadings with different pressure ranges and the inelastic deformations and vibrations are recorded. The training data is obtained by the midpoint displacements and the pressure acting on the plates. The signals are recorded by the measurement technique mentioned in [41]. The experiments are performed with aluminum plates having 553 mm diameter and 2 mm thickness as shown in Fig. 1a, while the steel and copper plates have a diameter of 138 mm with 2 mm thickness presented in Fig. 1b. The shock tube consists of a high-pressure chamber (HPC) and a low-pressure chamber (LPC) separated from each other by a membrane. The LPC has a length of 5.7 m and the length of the HPC can be arranged in between 1 and 2 m as shown in Fig. 1. The pressure difference between the HPC and the LPC causes the burst of the membrane resulting in a shock wave. This wave moves into the LPC and strikes the plate specimen at the end of the tube. The pressure evolution on the plate specimen during impulsive loading can be varied using different gases in the HPC. The lighter the gas such as helium, the faster the shock wave and, hence, the higher the pressure on the plate specimen. The time-dependent pressure along with other mechanical quantities mentioned in Sect. 3 are taken as the input for the training of the NN and the corresponding midpoint displacement is taken as the output.

## 2.1 Experimental and computational bottleneck

The Finite Element simulations of shock wave loaded plates were in accordance with the experiments as reported in [37]. These simulations, however, take a considerable amount of time to get completed. Thus, to have a faster way for approximating the midpoint displacements of shock wave loaded plates, a deep convolutional neural network was proposed in [28]. Although good results were obtained concerning the training data, the generalizing ability of the model was poor. In this research, we propose a new surrogate model comprising of gated RNNs and temporal convolutional network which perform efficiently on untrained validation experiments (The range of input values is beyond the range of training sequences).

The remaining article is dedicated to the development of a surrogate model which is computationally efficient for predicting the midpoint displacement.

## 3 Surrogate model for midpoint displacement prediction

The notation used in the current research is demonstrated as follows. The pressure acting on the plate is denoted by  $p$ , the stiffness relation  $s$  is obtained by dividing Young's Modulus by plate diameter, and the wave propagation velocity is denoted as  $v$ . The inclusion of stiffness relation in the study accounts for the distinct material and geometry of the plates, while the wave propagation significantly influences the deformation of shock wave-loaded plates. Additional inputs could also be utilized to characterize different nonlinear behavior of the materials tested. However, with the present study we utilize the internal parameters of the gated recurrent layers to implicitly learn the nonlinear behavior of different tested materials during the training process. This generates a nonlinear reduced order set of internal parameters responsible for tracing the nonlinear material response. For the case of TCN's, the dilated convolution ensures that the entire past information is taken into account without any information leak to encompass the desired path-dependent behavior. Also, apart from the materials used in the study if the deformation behavior for completely new materials

are need to be investigated then, in such a scenario the network must be re-trained with the corresponding stiffness relation to produce expected results. Thus, after introducing the variables, the notations used for sequence identification are as follows. The  $m$ th sequence is denoted by a superscript  $(m)$ , the  $i$ th step in a sequence is denoted by superscript  $\{i\}$ , an input sequence is represented as  $\mathbf{x}^{(m)} = ([p, s, v]^{(1)}, [p, s, v]^{(2)}, \dots, [p, s, v]^{(i)}, \dots, [p, s, v]^{(L)})$ , similarly the output sequence consisting of the midpoint displacement of a plate as  $\mathbf{y}^{(m)}$  and finally the  $m$ th ordered pair of input and output variables are represented as follows  $[\mathbf{x}, \mathbf{y}]^{(m)}$ . Further, to avoid convergence problems and have numerical stability we scale the input and the output variables. For each input and output feature denoted by  $\zeta$  the following operation is applied.

$$\zeta^n = \frac{\zeta - \zeta_m}{\zeta_d} \quad (1)$$

where,

$$\zeta_m = \frac{\zeta_{max} + \zeta_{min}}{2} \quad \text{and} \quad \zeta_d = \frac{\zeta_{max} - \zeta_{min}}{2} \quad (2)$$

In Eq. (2),  $\zeta_{max}$  and  $\zeta_{min}$  represent the maximum and minimum values of the feature under consideration and the operation in Eq. (1) maps the features to have the following range  $[-1, 1]$ . Thus, the scaled ordered pair of input and output sequences are represented as  $[\mathbf{x}^n, \mathbf{y}^n]^{(m)}$ . The proposed surrogate model which belongs to the class of NNs, thus non-linearly maps the normalized input sequence to the corresponding normalized output sequence.

$$\mathcal{M}_{NN}(\bar{\theta}) : \mathbf{x}^n \implies \tilde{\mathbf{y}}^n \quad (3)$$

where, the parameters of the model are given as  $\bar{\theta} = \{(\mathbf{W}_1, \mathbf{b}_1), (\mathbf{W}_2, \mathbf{b}_2), (\mathbf{W}_3, \mathbf{b}_3), \dots, (\mathbf{W}_L, \mathbf{b}_L)\}$ . The output of the surrogate model is essentially an approximation and it requires tuning the parameters such that the following equation is minimised.

$$e = \|\mathbf{y}^n - \tilde{\mathbf{y}}^n\|^2 \quad (4)$$

This process is referred to as training and is discussed in Sect. 4. Also, for simplicity, we drop the superscript  $n$  corresponding to the scaled variables and agree that only normalized quantities are provided for training and prediction through the NN. In the following section, the sampling procedure adopted for training the Neural Networks is presented.

### 3.1 Sampling

To train the surrogate models proposed in this section, we need to collect sequences consisting of pressure, stiffness relation, and wave propagation velocity. In this research, we do so by collecting experimental data corresponding to steel, copper, and aluminum plates. We realize that the experimental data does not cover all the possible deformation sequences that can occur and thus, we only work with a subspace of all possible deformations.

In the following, we now concretely build our sampling strategy. In this research, we consider experiments that are carried out over 15,000 time steps. To have a stack of experimental observations we analyze the sequences over 500 time steps, i.e., the input and the corresponding output vector are sequences having 500 time steps. These are extracted by employing a windowing technique, where a window of 500 time-steps is slid over the experimental observations to collect the required training data. Also, to avoid any flow of redundant information a stride corresponding to 500 steps is employed during the training process. Thus, a total of 100 sequences were extracted corresponding to the experiments of the three materials. They were further split into 75 training and 25 test sequences randomly. It should be noted that the patterns of both the input and the output sequences showcased in the training and test sequences are different. This helps in analyzing the performance of the trained NN models on patterns which are not provided during the training phase. Such a strategy is useful for scenarios where we have limited number of experiments and where each experiment consist of large number of time steps. This helps in avoiding extremely long training time and poor prediction abilities when entire set of experiments were used for training (15,000 time steps). Finally, the proposed models are tested across a series of validation experiments to determine which model is best suited for the study. The input sequences are stacked to form a variable  $\mathbf{x}$  and the output sequences are stacked to form  $\mathbf{y}$ . This procedure is repeated until all the timesteps in the experiments are covered to generate  $(\mathbf{x}, \mathbf{y})^{(T)}$ , where  $T = (1, 2, 3, \dots, N)$  unequivocally identifies the considered subspace of experiments. This concludes the sampling of the training data.

### 3.2 Neural network formulations

The most fundamental transformation available in neural networks is termed as the feed-forward or the fully connected transformation [42]. With the advent of deep learning, advanced layers like convolution, recursion have overcome the aforementioned shortcomings by parameter sharing [43] and adapting the topology to account for long time dependencies [17,35]. The difficulty of processing sequential data by fully connected networks can be overcome by using a Recur-

rent Neural Network (RNN) [44]. The RNNs make use of parameter sharing and recursions to keep trace of the memory while processing path-dependent data. However, there are couple of drawbacks that are associated with the simple RNN topology, those being the unstable gradients and receding memory [35,45]. This led to the development of gated recurrent neural networks such as the LSTM [35] and GRU [17]. These gated RNNs form the building block for the encoder–decoder architecture presented in Sect. 3.3. Further, since TCNs have showed better performance as compared to RNNs and with less number of parameters. In the present study, we develop a multilayered TCN network to compare its computational accuracy in comparison to other mentioned recurrent approaches.

### 3.3 Encoder–decoder architecture

Recurrent layer, Long-Short Term Memory and Gated Recurrent Unit neural networks are well established for sequence modeling, language modeling and machine translation [33,34,46]. In literature, a lot of efforts are undertaken to push forward recurrent language models with encoder–decoder architecture [47–49]. In the present study, we propose an encoder–decoder architecture with an intention of encoding a variable input sequence denoted as  $\mathbf{x}^{(m)} = ([p, s, v]^{(1)}, [p, s, v]^{(2)}, \dots, [p, s, v]^{(i)}, \dots, [p, s, v]^{(I)})$  into a fixed-length vector. This will be further processed by the decoder also comprising of gated recurrent layer and fully connected layer to generate a variable-length sequence of displacement vector  $\mathbf{y}^{(m)} = (d^{(1)}, d^{(2)}, \dots, d^{(i)}, \dots, d^{(I)})$ . Thus, the proposed model can be analogously considered as a neural mechanical translator that translates the mechanical quantities mentioned in the sequence  $\mathbf{x}^{(m)}$  to the corresponding midpoint displacement sequence  $\mathbf{y}^{(m)}$  of the plate. Subsequently, the limitation of a fixed-length representation of the input sequence is addressed [34] and a modified attention mechanism is presented in the following section which allows the model to automatically soft search for the relevant information in the encoded sequence to predict the target displacement. Like LSTMs/GRUs, CNNs also have the ability to process sequential data. However, it does not process input sequences of variable length inherently. This means, if we initialize a 1D-CNN layer to process 500 timesteps. An error message will be reported, every time the length of the sequences are less or more than 500 timesteps. To elaborate further, let us consider a case where the NN is deployed in a Finite Element simulation to compute the displacement behavior. Further, let us assume we are at the 100<sup>th</sup> time-step and we want to evaluate the solution at 101<sup>st</sup> time-step. The RNN based NNs maintain the sequence information through their internal variables and, hence, they would accept the internal variables from the previous time-step and the input at the current time-step to predict the displacement behavior

at the next time-step. However, for the CNN-based encoder–decoder architecture the entire loading sequence has to be passed i.e an input tensor with 101 time-steps to predict the correct displacement at the required 101<sup>st</sup> time-step. This implies, the input to the CNN based NNs will increase linearly requiring the entire history of sequences to predict the desired displacement. This would result in a computational bottleneck and hamper the speed of the simulation. Hence, although its easier to process and deploy CNN for encoder–decoder architectures, in the present research we employ LSTM/GRU cells because of the explicit maintenance of variables learning and following path-dependent solution incrementally. Since the GRU is a simpler variant of LSTM and shows comparable performance [50], we adopt it as the processor for extracting and exchanging information in the proposed framework. Also, the modified attention based encoder–decoder architecture consists of GRU and fully connected layers.

#### 3.3.1 GRU based encoder–decoder

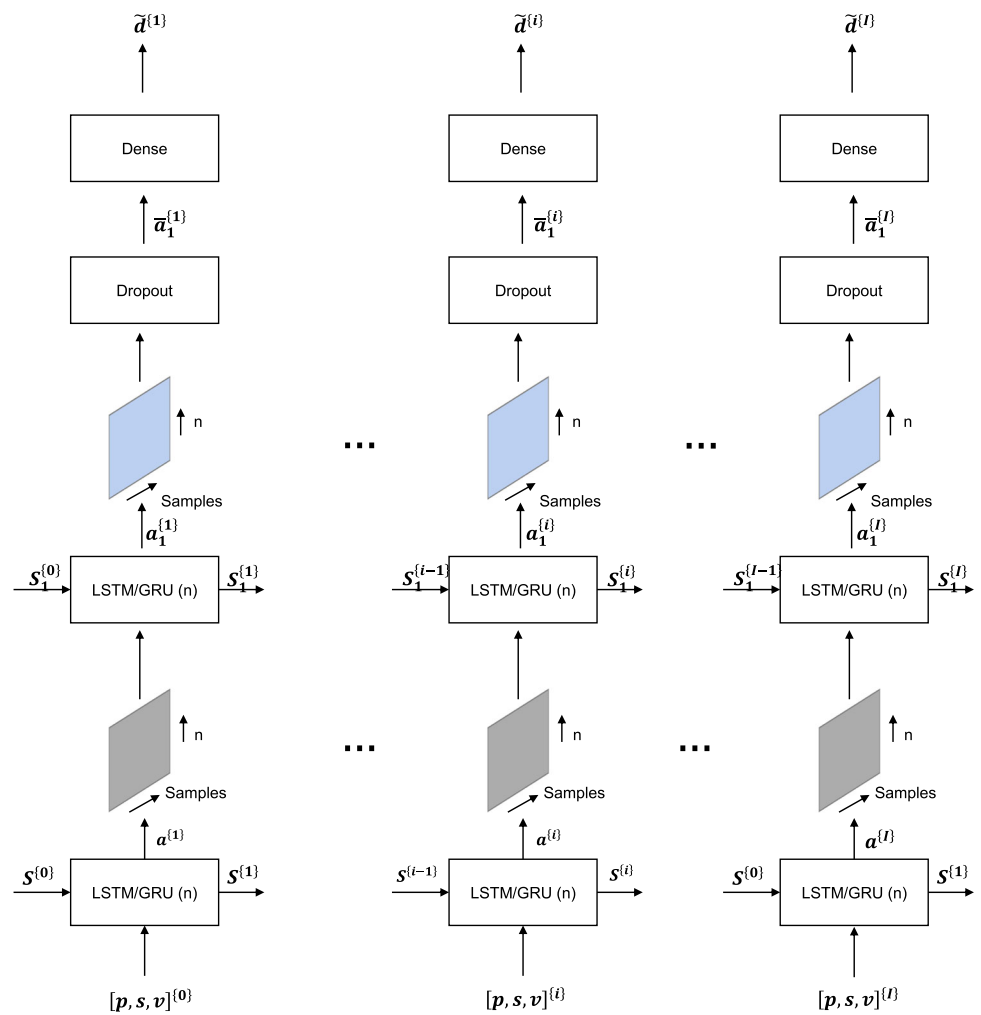
The proposed architecture outperformed the computational graphs mentioned in Fig. 2 especially concerning its predictions on the validation set (the limit of minimum and maximum values are beyond the training range), and the results are presented in Sect. 5. Before moving further, we highlight a crucial point of the architecture for encoding. As seen in Fig. 3, the input sequence consisting of  $I$  time steps is processed by the encoder, however, the task of memorizing the sequences and converting them into a fixed-length representation becomes difficult with an increase in sequence length. This can be a result of providing only the final hidden state of the encoder, making it difficult for the decoder to summarize long input sequences. With the present application of explicit dynamic analysis of shock wave-loaded plates, it becomes essential to process long sequences since the step size chosen for loading can be very small to maintain numerical stability. In a bid to overcome this problem, we introduce a modified attention mechanism that would enable the model to predict displacement at a time step  $i$  by automatically searching for the relevant information in the encoded sequence rather than paying equal attention to the entire sequence. Thus, in the section below the development of attention-based encoder–decoder architecture is proposed which outperformed all the models presented in Figs. 2, and 3.

#### 3.3.2 Attention based encoder–decoder architecture

The aim of the present study is to understand the applicability of the language processing topologies for path-dependent mechanical problems and to compare the performance not only with the gated recurrent neural networks but also with



**Fig. 2** Computational graph of combination of gated recurrent cells with fully connected layer: There are several nodes in the graph corresponding to two LSTM/GRU(n) layers where 'n' denotes the number of units of the cell, dropout and dense layer. The edges represents the operands corresponding to the input vector  $[p, s, v]^{(i)}$ , the state of the cell  $S^{(i)}$ , the output from the LSTM layers as  $a^{(i)}$  and  $a_1^{(i)}$ , the output from dense layers as  $\tilde{a}_1^{(i)}$  and the output vector  $\tilde{d}^{(i)}$ .



Temporal Convolutional network. It is clear from the studies stated above, that the internal variables of the gated recurrent layers are able to implicitly trace the evolution of path-dependent problems. However, the robustness of such architectures is a topic of ongoing research especially for mechanical problems. Since the encoder–decoder architecture with attention corresponding to translation problems were able to function robustly even for unknown input sequences, the exploration of its ability to handle/translate mechanical sequences is tested. The following modifications were applied on the encoder–decoder architecture to comply with the nonlinear response of the plate vibration.

1. An embedding layer at the interface of the information transfer between the attention module and the decoder is replaced by a learnable dense layer.
2. The loss function was changed from a probability driven categorical cross-entropy to mean squared error.

These modifications are essential and act as cross-road between language translation and mechanical response pre-

diction. In the proposed model, the encoder comprises of a GRU layer and is responsible for encoding the source sequence. Now, instead of passing the output of the encoder at the last time step  $I$ , we introduce modified attention that soft searches and utilizes the relevant information from the encoded sequence to decode an output at time step  $i$ . Mathematically this algorithm of soft searching is achieved by the following series of equations:

$$S^{(i)} = \tanh(W^{s1} o_e^{(m)} + W^{s2} h_1^{(i-1)} + b^s) \tag{5}$$

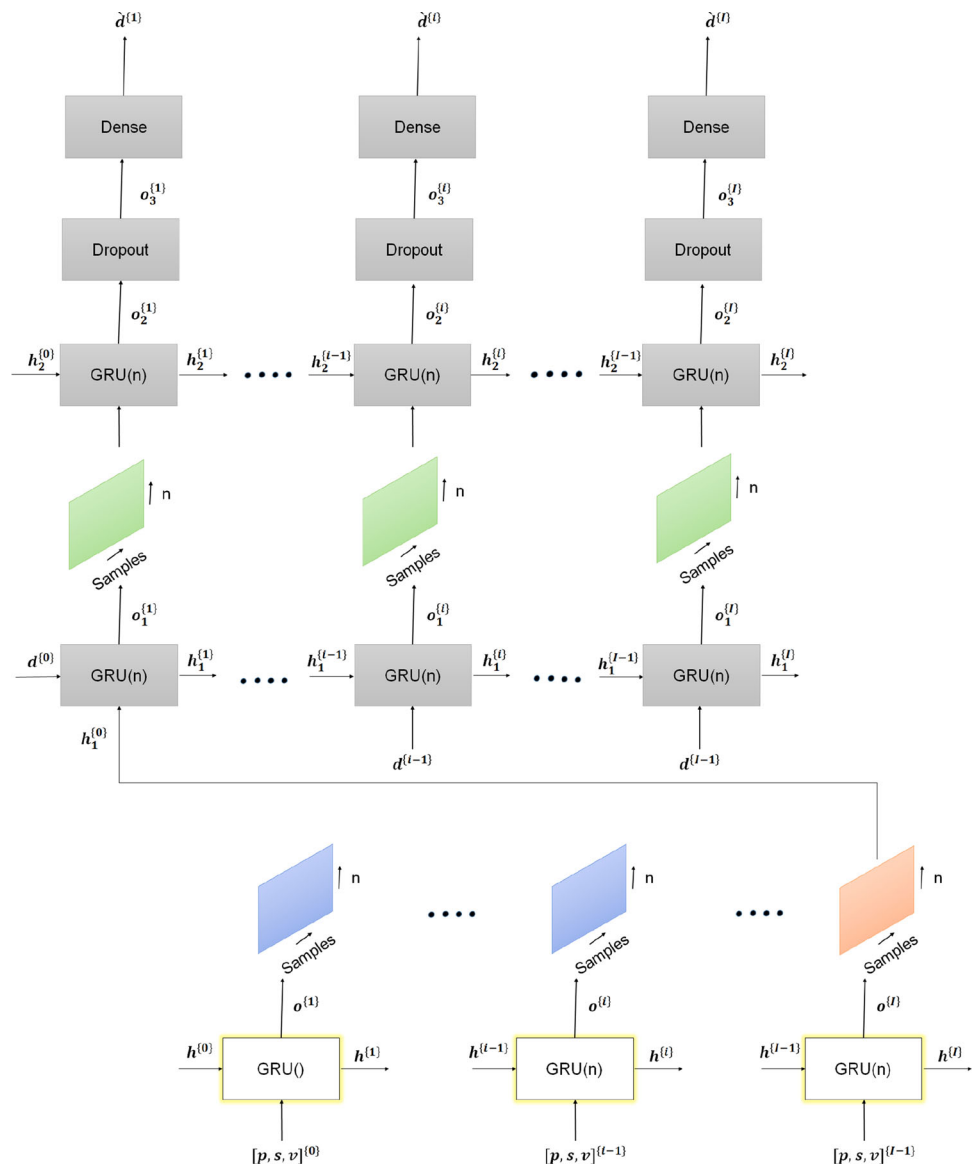
$$a^{(i)} = (V S^{(i)} + b^v) \tag{6}$$

$$\alpha^{(i)} = \frac{\exp(a_p^{(i)})}{\sum_{k=1}^l \exp(a_p^{(k)})} \tag{7}$$

$$C^{(i)} = \sum_{j=1}^l (\alpha \odot o_e^{(m)})_{pj} \tag{8}$$

Firstly, a score  $S^{(i)}$  between the hidden state of the decoder GRU layer  $h_1^{(i-1)}$  and the encoded sequence  $o_e^{(m)}$  is com-

**Fig. 3** Computational graph of employed encoder–decoder: The GRU transformation corresponding the encoder is highlighted by yellow boxes, the red output at the  $I^{\text{th}}$  time step of the encoder is passed to the decoder where the state variable  $h_1^{(0)}$  is initialised with it. All the transformations corresponding to the decoder are highlighted with gray boxes and  $n$  represents the number of units of the cell used. The edges represents the operands corresponding to the input vector to the encoder  $[p, s, v]^{(i)}$ , the state of the cell  $(h^{(i)}, h_1^{(i)}, h_2^{(i)})$ , input to the decoder  $d^{(i)}$ , the output from the GRU (decoder) layers as  $o_1^{(i)}$  and  $o_2^{(i)}$ , the output from dropout layer as  $o_3^{(i)}$  and finally the output vector  $\hat{d}^{(i)}$



puted following the procedure in [34]. This score is then passed to an alignment function, which is chosen to be the fully connected layer as highlighted in Eq. (6). This alignment function is responsible for learning and developing a relationship between the output of the decoder GRU  $h_1^{(i-1)}$  and the mentioned encoded sequence. Further, the parameters  $(W^{s1}, W^{s2}) \in \mathbb{R}^{n \times n}$ ,  $b^s \in \mathbb{R}^n$ ,  $V \in \mathbb{R}^{n \times t}$  and  $b^v \in \mathbb{R}^t$ , where  $n$  denotes the number of GRU units(cells) and  $t = 1$  denotes the number of output features, are optimized through backpropagation along with the entire model to compute a soft alignment. In the following step, an attention vector  $\alpha^{(i)}$  is computed by Eq. (7) which results in probability  $\alpha_q^{(i)}$  of the output  $o_1^{(i)}$  aligned to the source  $x^{(q)}$ . This is then used to compute the context vector  $C^{(i)}$  which extracts relevant information from the encoded sequence to predict the output  $o_1^{(i)}$ . Intuitively, this results in an attention mechanism in the

decoder and relieves the encoder from the burden of summarizing all the information from the source sequence into a fixed length representation.

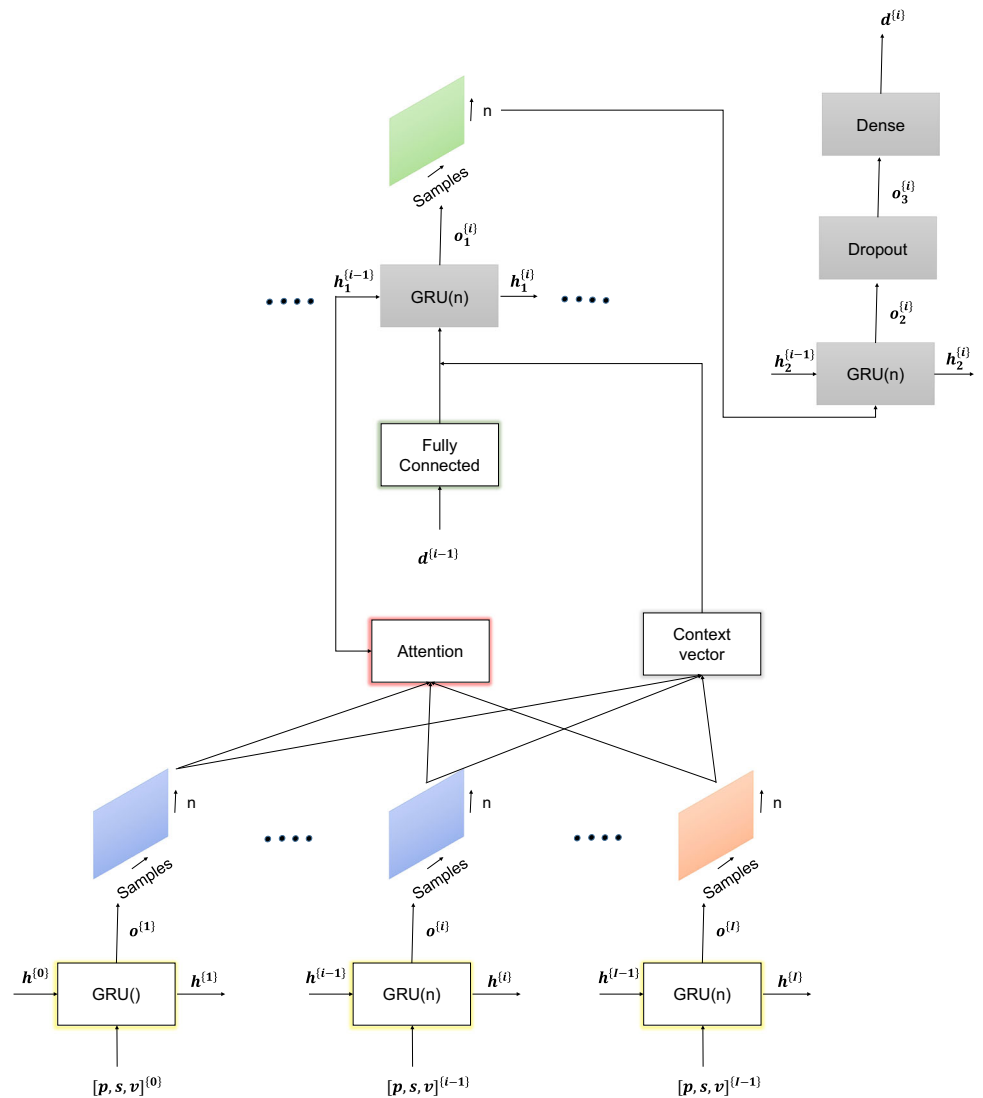
Until this point, the implementation of attention in decoder is similar as employed in the machine translation. In the following step, we introduce the modification of adding a fully connected layer instead of an embedding layer [51,52] for processing the input to the decoder. This is expressed as

$$f^{(i)} = W^f y^{(i-1)} + b^f, \tag{9}$$

with free parameters  $W^f \in \mathbb{R}^{n \times k}$  and  $b^f \in \mathbb{R}^n$ . This concludes the inclusion of attention in the decoder. Further, the context vector  $C^{(i)}$  and the vector  $f^{(i)}$  are concatenated and provided as an input to the decoder GRU layer. This is represented by

$$D_1 : [f^{(i)}, C^{(i)}] \mapsto o_1^{(i)}. \tag{10}$$

**Fig. 4** Computational graph of attention-based encoder–decoder: The GRU transformation corresponding to the encoder is highlighted by yellow boxes, the attention vector is computed by the transformation of encoded sequence and the decoder output at the previous time step to provide the probability values to the context vector which empowers the model with soft-search ability, the input to the decoder here is the concatenation of the output from the fully connected layer and the context vector. All the transformations corresponding to the decoder are highlighted with gray boxes and  $n$  represents the number of GRU cells. The edges represent the operands corresponding to the input vector to the encoder  $[p, s, v]^{(i)}$ , the state of the cell  $(h^{(i)}, h_1^{(i)}, h_2^{(i)})$ , input to the decoder  $d^{(i)}$ , the output from the GRU (decoder) layers as  $o_1^{(i)}$  and  $o_2^{(i)}$ , the output from dropout layer as  $o_3^{(i)}$  and finally the output vector  $\check{d}^{(i)}$ .



The operator  $\mathcal{D}_1$  is responsible for implementing all the operations that are included in the forward pass of the GRU layer followed by computing the output and updating its internal state  $h_1^{(i)} = o_1^{(i)}$ . Further as seen in Fig. 4, the output is processed by another GRU layer to enhance the capability of the model to learn from higher dimensional representation. The following equation represents this transformation mathematically:

$$\mathcal{D}_2 : o_1^{(i)} \mapsto o_2^{(i)} \tag{11}$$

Once, the output is computed, the state of the cell is updated  $h_2^{(i)} = o_2^{(i)}$  and this follows the processing through dropout layer. The dropout layer aims to introduce regularization and make the network realize patterns in the data. (The working of the dropout layer is explained in the following section). Finally, the output  $o_2^{(i)} \in \mathbb{R}^n$  is transformed to the space of the displacement sequence using a fully connected transfor-

mation expressed by

$$\mathcal{F} : o_2^{(i)} \mapsto \check{y}^{(i)} \tag{12}$$

In Eq. (12), the operator  $\mathcal{F}$  corresponds to the fully connected transformation mentioned with hyperbolic tangent as the scalar activation function and the output  $\check{y}^{(i)} \in \mathbb{R}^t$ , where  $t = 1$  represents the space of the midpoint displacement sequence mentioned above. This concludes the forward pass of the attention-based encoder–decoder architecture as presented in Fig. 4. With this model, we overcome the bottleneck of poor performance of the encoder with longer input sequences  $x^{(m)}$  and the results are presented in Sect. 5. Also, the adoption of soft search draws parallels to the FEM simulation of such shock wave-loaded plates as modeled in [37]. Since we encode the input sequence with  $I$  time steps at once and decode this information sequentially, suppose we are at



time step  $i - 1$  and we want to predict the displacement at the next time step, then the computed attention weights show more focus on the input  $\mathbf{x}^{(i-1)}$  and the loading history as compared to the remaining values of the encoded sequence. This development through attention-based models is analogous to the incremental procedure in FEM, where the displacement at the following step only depends on the input at the current time step and the previous history information. Also, in the proposed model the variables  $[\mathbf{h}^{(i)}, \mathbf{h}_1^{(i)}, \mathbf{h}_2^{(i)}]$  correspond to the internal variables of the encoder and decoder GRU layers which keep track of the evolution of the sequence. Thus, like mentioned in the previous sections, they form a reduced non-linear order reduction of the developed internal variables in a viscoplastic FEM analysis. The process inherited for training the proposed model is explained in Sect. 4. Finally, a dropout regularization technique [53] was implemented to make the computational graphs mentioned in Figs. 2, 3 and 4 learn the patterns present in the data rather than memorize the data itself.

### 3.4 Temporal convolutional network

The term temporal convolutional network (TCN), represents a descriptive term for a family of convolutional architectures and was first introduced in [29]. In the present study, we adopt the TCN proposed in [30], it has two main characteristics namely the convolution operation employed is causal in nature and secondly the network is capable of mapping an input sequence of an arbitrary length to the corresponding output having the same length.

For the accomplishment of the two mentioned characteristics, the 1D convolution operation is employed [54] along with the padding technique to ensure the length of the output is the same as the previous layer. The causal condition is satisfied by convolving only with elements from a certain time  $t$  and earlier time steps from the previous layer to generate an output at time step  $t$ . A major disadvantage of such a design is the need of either a very deep network or a large size of the filter to cover the entire historic data. This is overcome by adopting a dilated convolution which enabled an exponentially long receptive field [31]. More formally for a filter  $f$  and input sequence  $x \in \mathbb{R}^n$ . The dilation operation on an element  $e$  of a sequence at time step  $t$  is defined as

$$H(e) = \sum_{i=0}^{s-1} g(i) \cdot x_h \tag{13}$$

where  $s$  represents the size of the filter,  $i$  represents the level of the network and  $h = e - d \cdot i$  accounts for the path dependency, where  $d = 2^i$  in the present study represents the dilation base. Further, the receptive field of each layer can be computed through  $r = (s - 1) \cdot d$ , while for the complete residual block the following equation can be used.

$$r_f = 1 + 2 \cdot (s - 1) \cdot N_b \cdot \left( \sum_i d_i \right) \tag{14}$$

where,  $N_b$  represents the number of residual blocks. This gives us a clear indication of how to increase the receptivity of the layer i.e., either by choosing larger filter size  $s$  or increasing the depth of the network.

The deeper the network, the more important it becomes to stabilize the initialized model. In the present study, a residual block is used as proposed in [30]. It introduces a skip connection which provides an alternate path for the gradients to flow through the computational graph and enable smooth training process. The implementation of the residual block follows [55], and the computational graph employed in this study can be seen in Fig. 5. The TCN layer consists of one residual block ( $N_b = 1$ ), 32 filters, each filter having a size of  $s = 3$ . Since, higher number of residual blocks are recommended for sequences having large number of time-steps [55]. We increase the depth of the model by employing higher number of dilations  $d = [1, 2, 4, 8, 16, 32, 64]$ . This ensures that the receptive field of the model is more than 500 timesteps, which can be computed from Eq. (14)

## 4 Data processing and hyperparameter search

Prior to the training of the models proposed in the article the parameters are initialised randomly. Our aim is to tune these free parameters in a way that the NN learns the underlying function of the data. In order to do so, a loss function is employed which measures the error  $\tilde{e}$  between the predicted displacement sequence  $\hat{\mathbf{y}}^{(m)}$  and the sampled displacement sequence  $\mathbf{y}^{(m)}$  in the form

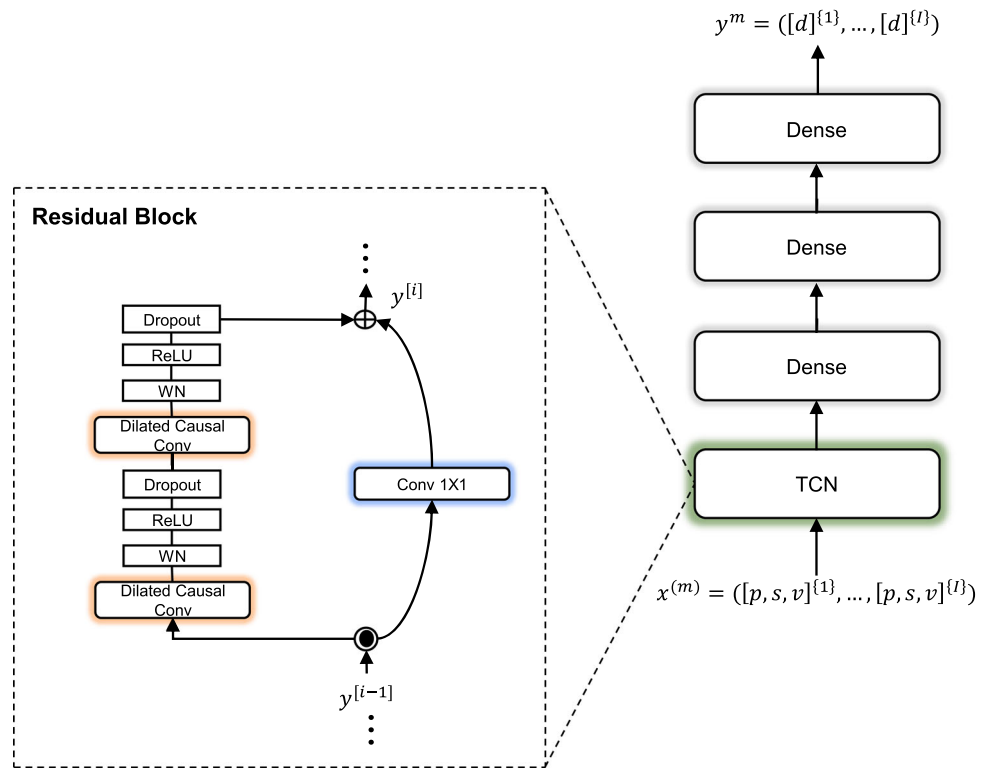
$$\tilde{e} = \frac{1}{N} \sum_{m=1}^N \mathbf{x}^{(m)} \text{ where, } \mathbf{x}^{(m)} = \frac{1}{l^{(m)}} \sum_{i=1}^{l^{(m)}} e^{(i)} \text{ and } e^{(i)} = \| \mathbf{y} - \hat{\mathbf{y}} \|^2 \tag{15}$$

Equation (15) is referred as the mean squared error, where  $N$  denotes the number of sequences used for training and  $l^{(m)}$  denotes the length of the  $m$ -th sequence. The trainable parameters of the computational graphs presented in Sects. 3.3 and 3.4 are optimized on the aforementioned loss function by a gradient descent optimizer. The optimizer used in research corresponds to the Adam optimizer [56].

### 4.1 Implementation details

The entire training process is implemented in Python. An in-house training algorithm for the modified attention steered

**Fig. 5** Computational graph corresponding to temporal convolutional network. The residual block is responsible for processing the history dependent information through levels of dilated convolution, while the integrated Dropout and Weight Normalization (WN) account for regularization of the graph. The input and the output through the graph are represented by  $x^{(m)}$  and  $y^{(m)}$



**Table 1** Neural network hyperparameters

NN architecture employed	Number of hiddenlayers	Number of GatedRNN or TCN layers	Number of units	Dense and other layers	Dropout(%)	Epochs
Sequential LSTM	4	2	160	1	15	1988
Sequential GRU	4	2	160	1	15	1143
Encoder–decoder without attention	5	3	128	1	10	4000
Encoder–decoder with attention	7	2	128	4	10	2500
Sequential TCN	3	1	32	3	10	3000

**Table 2** Hyperparameter search domain

Variable	Search domain
Number of LSTM/GRU layers	[1, 2, ..., 6]
Number of units	[16, 32, ..., 512]
Number of Dense layers	[1, 2, ..., 5]
Epochs	[0, 100, ..., 5000]
Number of units of TCN layers	[8, 16, ...128]
Dropout (%)	[5, 10, ...40]

**Table 3** Optimal parameters

NN architecture employed	Parameters
Sequential LSTM	309, 281
Sequential GRU	274, 273
Encoder–decoder without attention	232, 961
Encoder–decoder with attention	231, 810
Sequential TCN	61, 409

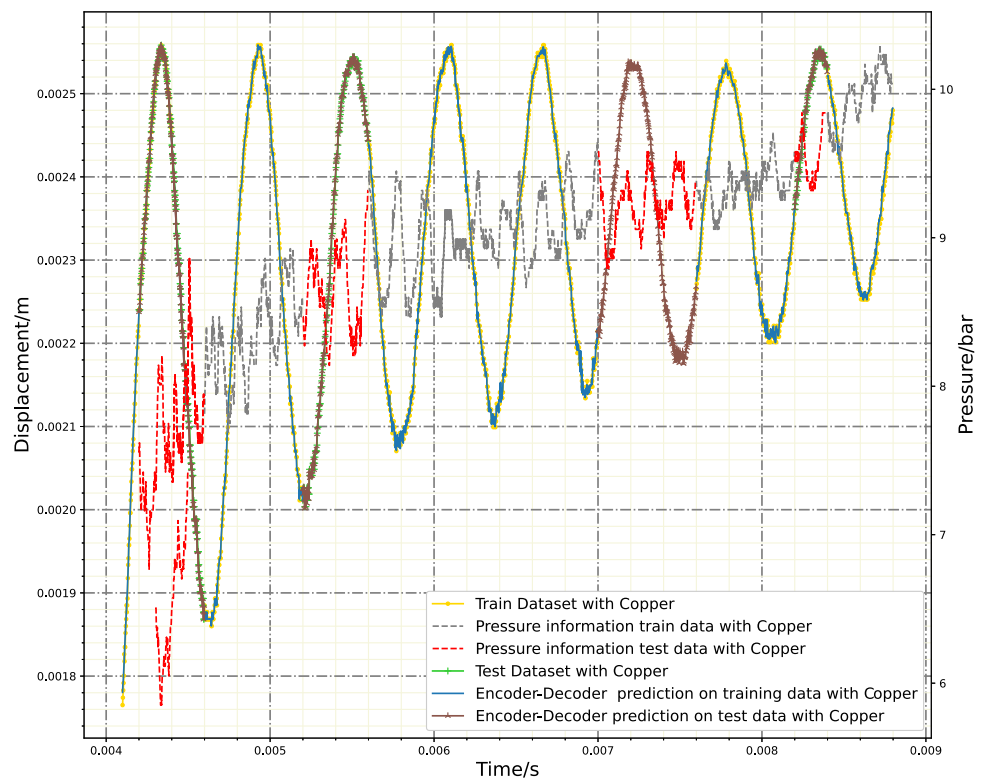
encoder–decoder is developed with TensorFlow library [57], while the remaining proposed models were trained with Keras library [58]. These computations were performed on a work station having 64 GB RAM, Nvidia GeForce RTX 3090 GPU and AMD Ryzen™ 7 5800 X CPU with 16 cores. Both

the neural network training and the prediction was performed on the GPU.

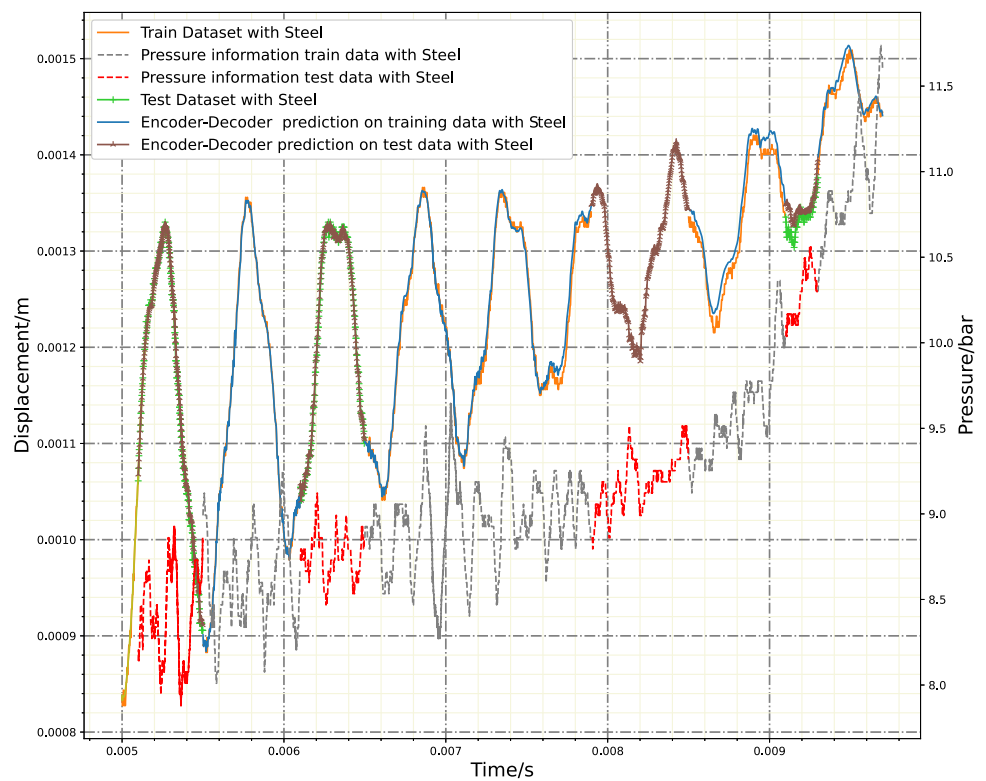
### 4.2 Tuning of hyperparameters

The parameters that cannot be trained by a gradient descent optimizer are commonly referred to as hyperparameters.

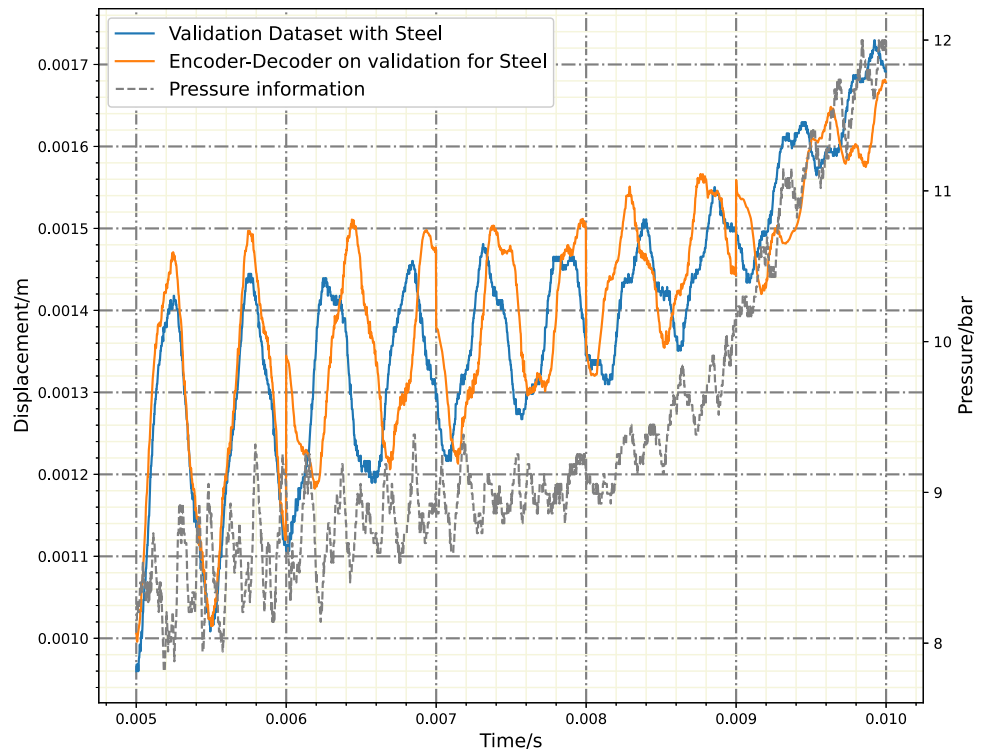
**Fig. 6** Combination of train-test measurements and NN simulations corresponding to encoder decoder topologies for a copper plate



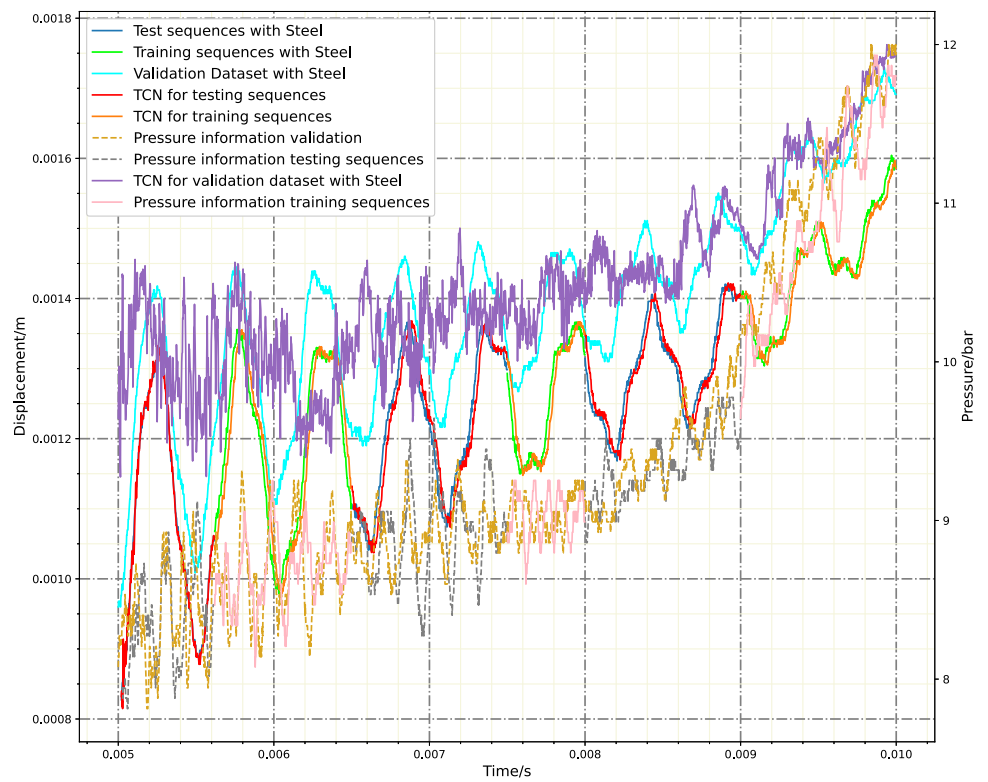
**Fig. 7** Combination of train-test measurements and NN simulations corresponding to encoder decoder topologies for a steel plate



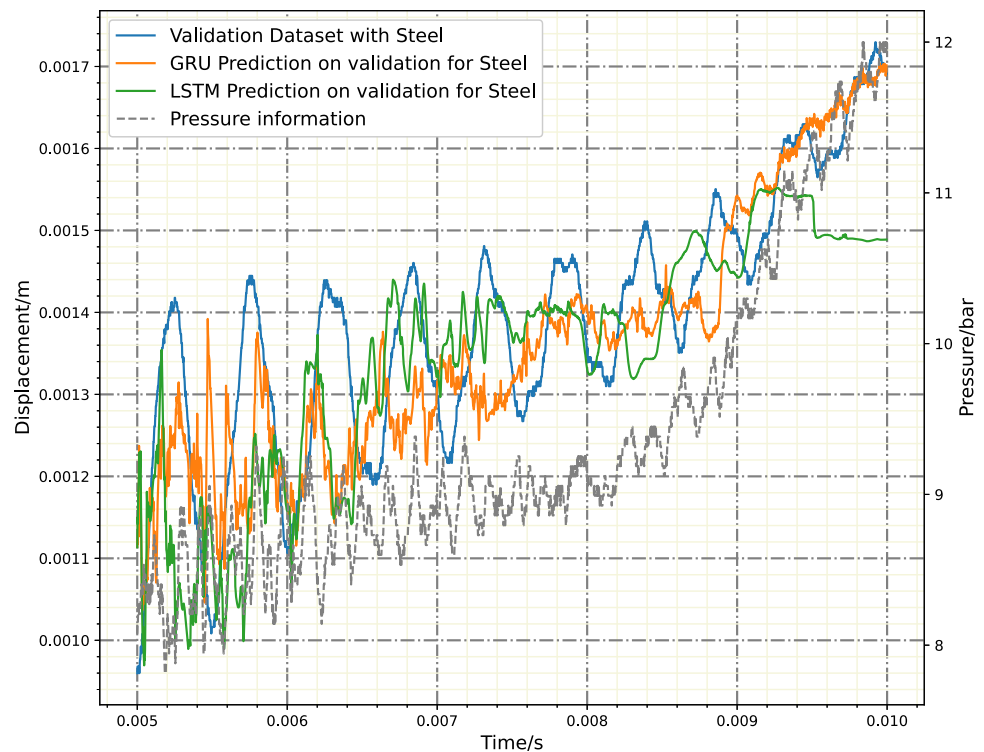
**Fig. 8** Validation measurements and NN simulations corresponding to encoder–decoder topology for a steel plate



**Fig. 9** Combination of train-test sequences, validation measurements, and NN simulations corresponding to TCN for a steel plate



**Fig. 10** Validation measurements and NN simulations corresponding to LSTM, and GRU topologies for a steel plate



These parameters are influential in the training process and the learning capabilities of the model, thus there is a need for tuning them. The hyperparameters that we tune in this research are the size of the LSTM/GRU/TCN cells ( $n$ ), the probability of dropout, and the number of gated RNN layers employed. Considering the experiences collected from previous studies with respect to the present physical problem, a maximum dropout rate of 0.4 [26,28] was used for the research. For the rest, we use typical hyperparameter values, for instance, the parameters of the Adam optimizer follow the recommendations mentioned in [56].

Systematic hyperparameter search methods constitute efficient solutions to find optimal hyperparameter configurations. Hyperparameter search algorithms provide an essential platform for automating the task of finding out optimal neural network configurations to achieve efficient performance. Since, the hyperparameters are usually interdependent and the effects it has on the model remain problem specific [59], it has popularized the implementation of search algorithms like the grid search [60]. Although grid search is effective in searching through the domain of the hyperparameter range, it is inefficient in its performance when large number of hyperparameters are involved. Random search algorithms [60], as the name suggests, explores statistically the entire search space by selecting a random combination of hyperparameter configuration with varying step sizes. This makes them suitable for hyperparameter-intensive neural networks [60]. However, the computational effort remains high.

In the present study, to utilize the advantages of the random search algorithm and make the search strategy computationally feasible we employ a Hyperband search algorithm [59]. Hyperband combines efficient exploration and investigation of suitable architectures and eliminates the requirement of any additional hyperparameters to be tuned. Independent of the search algorithm, an early stopping method was introduced which monitors the moving averages of the test-split sequences, these sequences represent a small subset (30%) of the training sequences. This results in an increase in the efficiency of the search strategy employed for the research problem and merits the choice of the hyperparameters for further investigations. Finally, the search domain and the hyperparameters used for successful implementation of the proposed NN models are as shown in Tables 1, 2 and 3 respectively.

### 4.3 Scaling strategy to predict results on validation experiments

It is well documented that NNs perform efficiently on the training data and can interpolate accurately within the training range [61]. In this study, we propose a scaling strategy that would empower the trained NN to predict data that has its limits i.e., maximum and minimum value of the dataset beyond the limits of the training range.

The scaling of the input and the output variables is performed by the linear transformations presented in Eqs. (1) and (2). This results in the range of the data between  $[-1, 1]$



and assists the NN to have a stable training process. In this study, we refer to the scaled input and output as the variables being in the normalized range. Once, the network is trained with desired accuracy and strategy presented in Sect. 4, it is deployed to make predictions on validation experiments. In this research, the validation experiments have the maximum and minimum limits beyond the training range (extrapolation) as mentioned above. Here, we highlight a different way of handling the validation data. Firstly, we scale the validation input sequence by computing the Eqs. (1) and (2) followed by the forward pass through the trained NN. This results in the output being in the normalized range, thus one must do an inverse transformation to bring the values back to the physical range. This is performed by the following inverse transformation:

$$\zeta = \zeta^n \cdot \zeta_d + \zeta_m \quad (16)$$

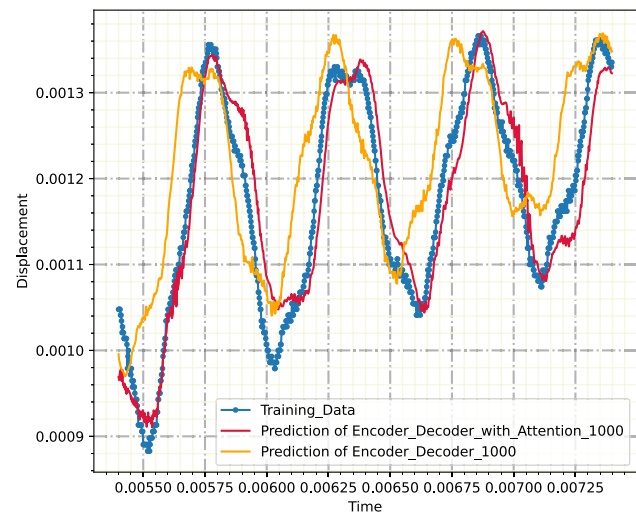
$$\text{where, } \zeta_m = \frac{\zeta_{max} + \zeta_{min}}{2} \quad \text{and} \quad \zeta_d = \frac{\zeta_{max} - \zeta_{min}}{2} \quad (17)$$

While transforming the output to the physical range for validation experiments it is important that the approximate range in the physical range is known to the user using the surrogate model [25,26], because the scaling variables used for training the NN has a different operating range and hence utilizing them would not lead to a correct physical representation of the displacement sequence. This, means the output from the NN can be scaled to any physical range provided the patterns of the input sequence do not starkly differ from the training sequence and the user through his experience of experiments can predict an operating range. We want to highlight this ability of the NN i.e., being scalable for learned patterns in the data enabling it to handle validation sequences having different physical ranges. Finally, we would also like to report that the accuracy of the results in the physical space are highly dependent on the range of the physical displacement chosen by the user working with the NN model. This puts an additional requirement on the user to be thorough with the physical evolution of the dynamic behavior to avoid large deviations of the results in the physical space when dealing with sequences having ranges outside of training range.

Adopting this technique along with the proposed modified attention-based encoder–decoder architecture yielded the best results over the validation experiments and is presented in Sect. 5.

## 5 Results and discussions

All the values of the graph presented below correspond to values in the physical range. In the experiment with steel plates the recorded time range is 5 ms, for copper plates it corre-

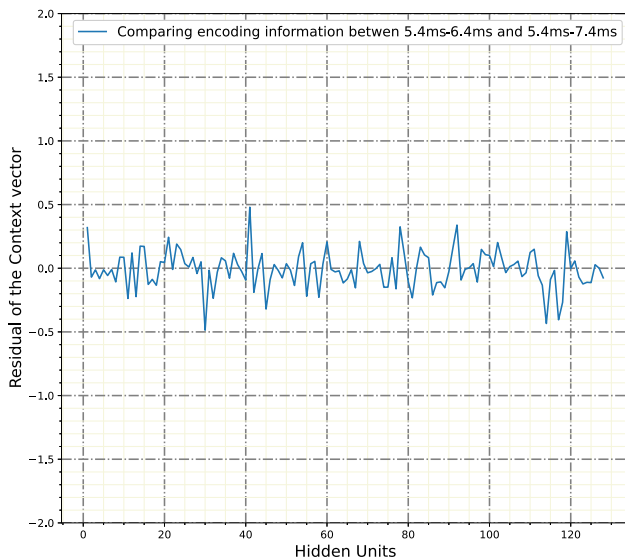


**Fig. 11** Prediction of the encoder–decoder model with and without Attention over the experimental measurements

sponds to 5.3 ms and for aluminium plates it is 7.7 ms. The experimental setup for copper and steel plates can be seen in Fig. 1. Peak pressure was generated at the start of the experiment through the utilization of helium in the HPC to generate a shock wave. The pressure generated through the experiments along with the stiffness relation and wave propagation velocity was used to train the computational graphs depicted in Figs. 2, 3, 4 and 5. Further, to have better visualization of the results corresponding to the train and test sequences, we present the result corresponding to only encoder–decoder topologies (see Figs. 6, 7, 13). An overview of responses of the other topologies can be seen in Table 5. The predictions of the NN over the mentioned input sequence for copper are depicted in Fig. 6.

The RMSE values shown in Table 5 showcase that all the proposed topologies were able to closely learn the dynamic response of a viscoplastic nonlinear analysis of a copper plate. Furthermore, a similar result was observed in Fig. 9 for the prediction of the NN over the pressure sequence used to dynamically load a steel plate. The interpolation behavior of these models are examined by splitting the data randomly to training and testing sets and letting the NN learn through the training data. The response of the trained NNs on the test sequences highlights its ability to implicitly learn the function, instead of memorizing the data itself.

Though all the models learn to approximate the dynamic response of the shock wave-loaded plate, there is a major difference in the way in which the input information is processed by the four models. In the computational graphs displayed in Figs. 2, and 5 the information is processed sequentially i.e., a transformation is directly learned to map an input sequence to a corresponding output sequence through the trained parameters of the NN model. However, in the case of



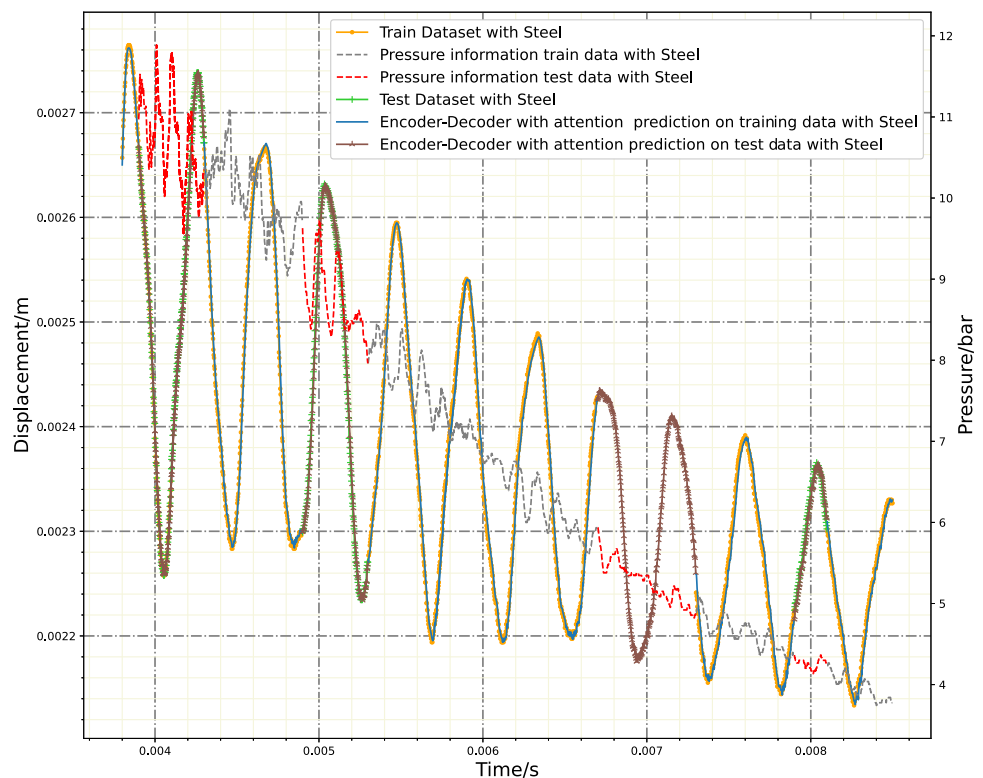
**Fig. 12** Comparison of the residual of the encoded information between the time-steps 5.4–6.4 ms which corresponds to 500 time-steps and encoded information between 5.4 and 7.4 ms which corresponds to 1000 time-steps for attention framework

encoder–decoder architecture, the input information is summarized in a fixed-length vector called the context vector, while the encoder output is discarded. This context vector aims to encapsulate all the relevant information corresponding to the input sequence, providing a concise representation

of the pressure history used by the decoder to translate it into a displacement sequence.

To further investigate the effectiveness of the proposed models, we observe their predictions over validation input sequences (see Figs. 8, 9, 10, and 14). These experiments have similar data patterns but have different physical maximum and minimum feature limits. All the validation sequences are extracted from experiments that are completely anonymous to the trained models and the response to the input data is evaluated on the entire experiment. This helps in evaluating the robustness of the surrogate models over unseen data and assists us in choosing the model performing the best. It is clear from the Figs. 8, 9, 10 that between the time range of 5 ms to 8.8 ms the proposed encoder–decoder model completely outperforms the other sequential LSTM, GRU and TCN models, portraying a robust nature of the mechanical translation achieved. The sequential models, though trained with dropout regularization could not achieve a stable response over the validation input sequence. This highlights how critical and effective is the information embedded in the context vector which imbibes the intricate information of the validation pressure sequence and passes it to the decoder to generate a stable displacement response. Also, it must be noted that the current output from the decoder is generated in parts, with the input information being encoded every 1 ms to generate the displacement response. Further, from 8.8 to 10 ms the sequential models show improved performance matching the prediction of the decoded sequence.

**Fig. 13** Combination of train-test sequence measurements and NN simulations corresponding to encoder decoder with attention topology for a steel plate



**Table 4** Comparable parameters

NN architecture employed	Parameters	RMSE (Fig. 14)
Sequential LSTM	232, 985	0.60136
Sequential GRU	232, 961	0.421455
Encoder–decoder without attention	232, 961	0.0793
Encoder–decoder with attention	231, 810	0.03553
Sequential TCN	233, 715	0.5445

This can be because of the similarity of the validation and the training pressure information in the mentioned range, indicating the high sensitivity of the sequential models over the input sequence. Though, the fixed representation of the context vector has certain advantages as highlighted above. The same representation can be a hindrance to capturing variable-length long-term dependencies. Since the encoder–decoder framework compresses all the information from the input sequence, the probability of losing credible information with a long sequence increases naturally affecting the prediction of the decoder. When the input sequences are small as in the case of machine translation, the possible information loss is small and can be restored by the decoder keeping the semantics intact.

However, for the current research problem which is path-dependent and consists of long historical data the information loss can become astounding due to the compressed representation of the context vector, whose dimension is limited by the hidden state of the GRU encoder. Such a behavior can be observed in Fig. 11, where the yellow curve corresponds to the decoded sequence having an input length of 1000 points which corresponds to a time range of 2 ms. To further enhance the learning capability of the encoder–decoder framework, we have introduced an attention mechanism to bolster its capability of encoding the input information.

The concept of Attention has become increasingly popular in training the NNs with data having super long term dependencies [34]. This approach works by exploring the relevant information from the encoder output and its hidden state at the current time step to generate the prediction at the next time step. This helps the framework in exploring the inter-dependencies across the input and output space, which eventually contributes to recognizing the most influential historical state. The improved response of the framework after including attention can be seen in Fig. 11 where even after increasing the length of the encoded sequence we were able to obtain a stable response from the framework. On the flip side, if we look at the response of the framework without attention we can see that due to the increase in the length of the encoded sequence a compression loss of around 0.75 ms takes place. This can be seen clearly from the yellow curve where it generates a pattern at 5.25 ms which it should have

generated at the time 6 ms. This indicates that it uses the input information from 6 ms and it could not retrieve the information before it. To further investigate this behavior, we visualize the context vector representation with varying lengths of the input sequence.

In Fig. 16a, b, the residuals of encoded representations between the time-steps 5.4–6.4 ms and 6.4–7.4 ms are compared with an encoded representation between 5.4 and 7.4 ms. It is evident from both figures that, although the encoded sequence with more time-steps consist of the input information from 5.4 to 7.4 ms, the deviation in the time domain 5.4–6.4 ms is much higher (RMSE = 0.5490) in comparison to the time domain presented in Fig. 16 which has a RMSE value of 0.1487. This means even though the encoded sequence was supposed to contain a combined representation of the context vector between 5.4 and 7.4 ms, it only showed a representation corresponding to the time range 6.4–7.4 ms. This highlights the information loss that takes place in our encoder–decoder framework without attention

The context vector representation with attention follows another principle. It must be noted that because of the inclusion of the attention mechanism, the context vector is generated at every time step instead of it being encoded only once as in the case of our framework without attention. Such an enhancement empowers the decoder to soft search the required information from the encoded output and the decoder output at the current time step to give a prediction at the following time step. In Fig. 12, the residual of the context vector representation is portrayed for sequences between the time-steps 5.4–6.4 ms and 5.4–7.4 ms. It can be concluded from the figure that, in comparison to framework without attention (see Fig. 16a), the model with attention enhances the ability of the model to retain the encoded information (RMSE = 0.14877) even though the length of the encoded sequence increases in comparison to training length. This results in an effective and a robust model for tackling explicit dynamic problems.

Another deformation concerning a steel plate is presented in Fig. 13. It can be observed by comparing the Figs. 6, 7 and 13 that the pressure impulse follows a completely different trend. Since the NNs have an inherent capability to approximate and learn different patterns from the data, the pressure information presented in Fig. 13 was used to train and test the network to learn the corresponding deformation behavior. It can be observed that, for both the train and test sequences not only the model with attention was able to perform the best, but also for the validation data a stable response was obtained only for the proposed encoder–decoder framework with attention (see Fig. 14a, b). This is a further demonstration of the robustness of the framework including attention for complex nonlinear prediction.

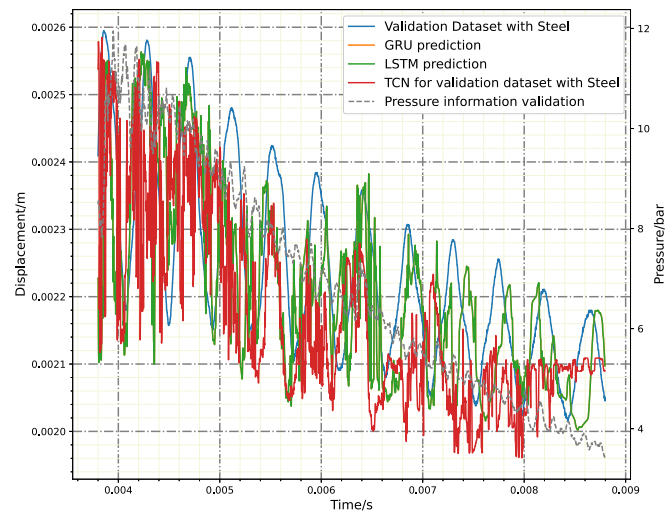
A summary of the performance of all the proposed five models is presented in Table 5. All the models per-

**Table 5** Comparison of root mean squared error (RMSE) between measurements and NN predictions

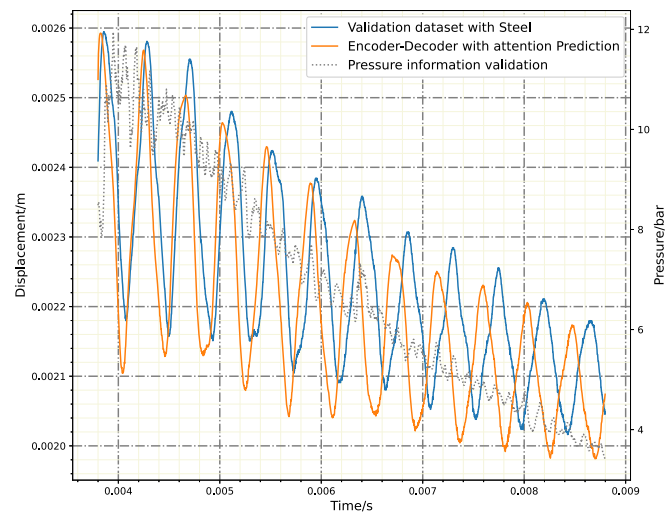
Measurement and NN prediction	LSTM (Fig. 2) RMSE	GRU (Fig. 2) RMSE	Enc–Dec without attention (Fig. 3) RMSE	Enc– Dec with attention (Fig. 4) RMSE	Sequential TCN (Fig. 5)
Deformation (Fig. 6) (T)	0.0178	0.018911	0.0231	0.01188	0.01667
Deformation (Fig. 7)(T)	0.011824	0.02245	0.0011	0.000881	0.0008981
Deformation (Figs. 8, 10, 9)(V)	0.2521	0.2392	0.0316	0.02981	0.2185
Deformation (Fig. 13) (T)	0.0289	0.026	0.00587	0.001391	0.00178
Deformation (Fig. 14) (V)	0.27	0.31	0.0793	0.03536	0.289

The samples corresponding to training and test are indicated by (T), while the validation samples are indicated by (V)

**Fig. 14** Analysing the behavior of the proposed models for validation measurements



(a) Validation measurements and NN simulations corresponding to LSTM, GRU and TCN topologies.



(b) Validation measurements and NN simulations corresponding to Encoder-Decoder architecture.

**Fig. 15** Combination of train-test sequence measurements, validation measurements and NN simulations corresponding to encoder decoder with attention for an aluminium plate

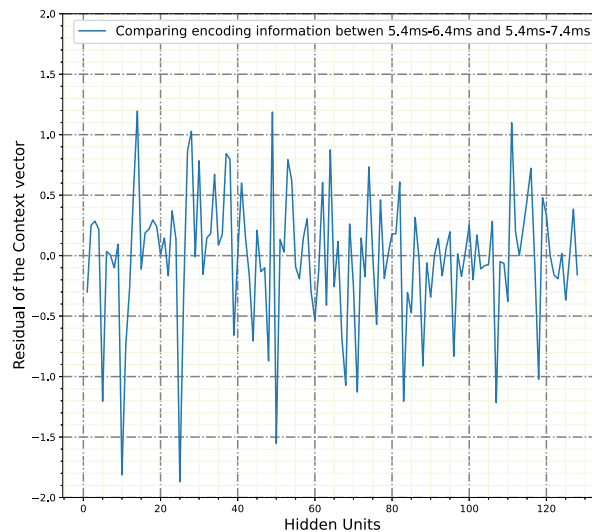


form relatively well on training data as highlighted by the RMSE values for Figs. 6, 7, 9 and 13 indicating that the models approximated the required underlying function. A special mention has to be given for the TCN architecture which performed either better or on par with the sequential models initialized with gated recurrent networks with less parameters. However, the proposed encoder–decoder framework with attention outperforms the remaining models for its response on validation sequences boasting a minimum RMSE value of 0.02981 for its response to a steel plate deformation. Further, in Table 4, the RMSE values corresponding to models consisting of similar number of parameters are presented. Since, the encoder–decoder framework shows better performance in comparison to other sequential models. We initialize new sequential models to have similar parameters as the encoder–decoder framework. The results after training the models are evaluated on a validation sequence presented in Fig. 14 to compare and understand whether the stable performance of the attention framework should be attributed to the number of parameters with which it is enriched or the way they process information through the architecture. As it can be seen from Table 4, although the models were freshly initialized and trained with comparable parameters, the model with attention framework performed the best for validation sequence presented in sequence Fig. 14. Also, it is interesting to highlight that the newly initialized and trained LSTM, GRU and TCN models performed less efficiently than their counterparts with optimal parameters, which was expected.

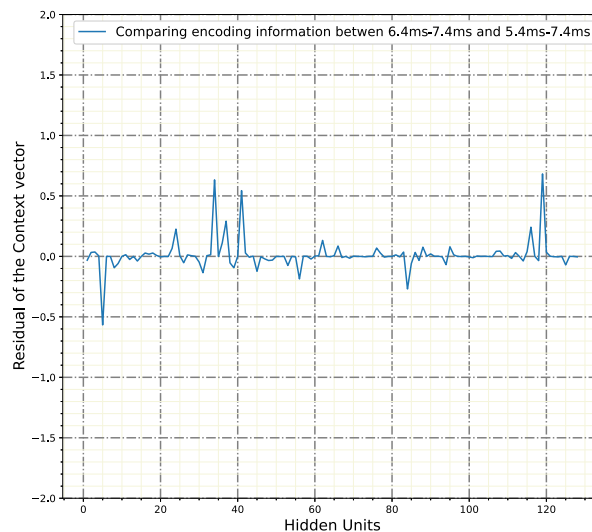
In Fig. 15, two aluminium plate deformations, measured in the large shock tube from Fig. 1 are shown. For the deformation data used for training, a blue displacement curve highlighted in Fig. 15 is used. An aluminum membrane with Nitrogen was used in the HPC to generate its response, while for the validation highlighted by the red curve an aluminum membrane with Helium was used. The desired deformation portrayed large deflections, instead of exhibiting oscillations as in the case of steel and copper plates. The framework with attention was able to understand and learn the response of the aluminum plate not only for the train and test sequences but also for the validation sequence. Further, if we look at the test sequence highlighted by green font we realise that no information from this time range is utilized for training the NN model and yet the NN was able to predict the nonlinear evolution in this region. This further reiterates the robust nature of the proposed framework for predicting different deformation patterns.

Finally, it must be noted that in the present study the models approximate both geometrical and physical nonlinearities through training data and thus belongs to the class of a data-driven or data-dominant approach. In such a scenario, the focus is to learn the underlying function through the data or to learn the patterns present in the data. If the material under consideration is expected to show different deformation evolution in comparison to Steel, Aluminium and Copper, then re-training of the model becomes necessary to efficiently predict the deformation evolution corresponding to the material.





(a) Comparison of the residual of encoded information between the time-steps 5.4ms–6.4ms which corresponds to 500 time-steps and between 5.4ms–7.4ms which corresponds to 1000 time-steps.



(b) Comparison of the residual of encoded information between the time-steps 6.4ms–7.4ms which corresponds to 500 time-steps and between 5.4ms–7.4ms which corresponds to 1000 time-steps.

**Fig. 16** Analysing the context vector of the encoder–decoder framework

## 6 Conclusions

In the present study, we evaluated firstly gated RNN models which include LSTM and GRU cells followed by a TCN architecture to develop an efficient surrogate model to learn the mid-point deformation behavior of complex path-dependent shock wave loaded plates. Secondly, we train and evaluate an encoder–decoder topology with gated RNNs (similar to language translation) to understand and explore

the way of approximating sequential mechanical problems. Subsequently, we enhanced the proposed encoder–decoder architecture with attention weights (soft-search) to significantly improve the generalizing capabilities of the surrogate model. All the mentioned sequential models were trained, using the data that was collected experimentally and through the proposed sampling procedure. Finally, the models were assessed on training and validation experiments and it was concluded that the proposed encoder–decoder architecture

with attention performed the best for training and validation experiments.

The proposed model showed accurate predictions on training and validation experiments, however, it must be noted that this may not be always the case. The accuracy of the trained surrogate depends on the patterns that it learns through the training data. Thus, it may be required to retrain the model in case of learning new patterns of deformation behavior. The next question that we address is the interpretation of the developed internal variables in all the proposed models. Since, the surrogate models learn the deformation behavior for shock wave loaded plates, as long as the model is accurate the internal variables of the gated RNNs can reproduce the nonlinear response and the physical features are reproduced by the trained model. Furthermore, the attention weights also intuitively assign a physical resemblance to the decoder which pays maximum attention to the pressure information at the current time step and assists the decoder to produce the displacement information at the corresponding time step. This is similar to the Finite Element Method where the displacement is also computed sequentially.

The trained model with attention performs very well within the training range and interpolates precisely. With this study, we take a step further and present a simple scaling strategy that empowers the model to predict accurate results for input beyond the training range. However, it must be noted that the NN can only predict beyond the training range if the patterns learned by the network are similar to the expected displacement behavior. In any other case, it must be required to retrain the model with new data.

The main limiting factor of the proposed strategy is that the NN is trained on specific data patterns and for certain plate geometries. To circumvent this issue, the current research consisting of the proposed encoder–decoder framework with attention could be employed for intelligent stiffness computation. Since, the implementation in Finite Element framework also requires understanding the way the NN learns and behaves on unseen data. The present research, aims towards providing an explainable learning strategy of the presented encoder–decoder framework which can be utilized in a more generalized manner with the Finite Element Method.

**Acknowledgements** The authors gratefully acknowledge the financial support provided by Deutsche Forschungsgemeinschaft (DFG: Grant No. STO 469/17-1).

**Funding** Open Access funding enabled and organized by Projekt DEAL.

**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. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. *Nature* 521:436–444. <https://doi.org/10.1038/nature14539>
2. Shakiba M, Parson N, Chen X-G (2016) Modeling the effects of cu content and deformation variables on the high-temperature flow behavior of dilute Al–Fe–Si alloys using an artificial neural network. *Materials* 9(536):1–13
3. Liu G-R, Xu Y-G, Wu Z-P (2001) Total solution for structural mechanics problems. *Comput Methods Appl Mech Eng* 191:989–1012
4. Waszczyszyn Z, Ziemiański L (2001) Neural networks in mechanics of structures and materials—new results and prospects of applications. *Comput Struct* 79:2261–2276
5. Stoffel M, Bamer F, Markert B (2019) Stability of feed forward artificial neural networks versus nonlinear structural models in high speed deformations: a critical comparison. *Arch Mech* 71(2):95–111. <https://doi.org/10.24423/AOM.3091>
6. Heider Y, Wang K, Sun W (2020) So(3)-invariance of informed-graph-based deep neural network for anisotropic elastoplastic materials. *Comput Methods Appl Mech Eng* 363:112875. <https://doi.org/10.1016/j.cma.2020.112875>
7. Thaler D, Stoffel M, Markert B, Bamer F (2021) Machine-learning-enhanced tail end prediction of structural response statistics in earthquake engineering. *Earthq Eng Struct Dyn*. <https://doi.org/10.1002/eqe.3432>
8. Hambli R, Katerchi H, Benhamou C-L (2011) Multiscale methodology for bone remodelling simulation using coupled finite element and neural network computation. *Biomech Model Mechanobiol*. <https://doi.org/10.1007/s10237-010-0222-x>
9. Chaboche J-L (1981) Continuous damage mechanics—a tool to describe phenomena before crack initiation. *Nucl Eng Des* 64:233–247. [https://doi.org/10.1016/0029-5493\(81\)90007-8](https://doi.org/10.1016/0029-5493(81)90007-8)
10. Stoffel M, Bamer F, Markert B (2019) Neural network based constitutive modeling of nonlinear viscoplastic structural response. *Mech Res Commun* 95:85–88
11. Stoffel M, Bamer F, Markert B (2018) Artificial neural networks and intelligent finite elements in non-linear structural mechanics. *Thin Walled Struct* 131:102–106. <https://doi.org/10.1016/j.tws.2018.06.035>
12. Lu X, Giovanis DG, Yvonnet J, Papadopoulos V, Detrez F, Bai J (2019) A data-driven computational homogenization method based on neural networks for the nonlinear anisotropic electrical response of graphene/polymer nanocomposites. *Comput Mech* 64:307–321. <https://doi.org/10.1007/s00466-018-1643-0>
13. Theocaris PS, Panagiotopoulos PD (1995) Plasticity including the Bauschinger effect, studied by a neural network approach. *Acta Mech* 113:63–75. <https://doi.org/10.1007/BF01212634>
14. Theocaris PS, Panagiotopoulos PD (1997) On the parameter identification problem for failure criteria in anisotropic bodies. *Acta Mech* 123:34–56. <https://doi.org/10.1007/BF01178399>
15. Ghavamian F, Simone A (2019) Accelerating multiscale finite element simulations of history-dependent materials using a recurrent neural network. *Comput Methods Appl Mech Eng*. <https://doi.org/10.1016/j.cma.2019.112594>
16. Wu L, Nguyen VD, Kilingar NG, Noels L (2020) A recurrent neural network-accelerated multi-scale model for elasto-plastic heteroge-

- neous materials subjected to random cyclic and non-proportional loading paths. *Comput Methods Appl Mech Eng* 369:113–234. <https://doi.org/10.1016/j.cma.2020.113234>
17. Chung J, Gulcehre C, Cho K, Bengio Y (2014) Empirical evaluation of gated recurrent neural networks on sequence modeling
  18. Gorji MB, Mozaffar M, Heidenreich JN, Cao J, Mohr D (2020) On the potential of recurrent neural networks for modeling path dependent plasticity. *J Mech Phys Solids* 143:103972. <https://doi.org/10.1016/j.jmps.2020.103972>
  19. Tandale SB, Bamer F, Markert B, Stoffel M (2022) Physics-based self-learning recurrent neural network enhanced time integration scheme for computing viscoplastic structural finite element response. *Comput Methods Appl Mech Eng* 401:115668. <https://doi.org/10.1016/j.cma.2022.115668>
  20. Kirchdoerfer T, Ortiz M (2016) Data-driven computational mechanics. *Comput Methods Appl Mech Eng* 304:81–101. <https://doi.org/10.1016/j.cma.2016.02.001>
  21. González D, Chinesta F, Cueto E (2019) Thermodynamically consistent data-driven computational mechanics. *Continuum Mech Thermodyn* 31:239–253. <https://doi.org/10.1016/j.cma.2016.02.001>
  22. Ladevèze P, Néron D, Gerbaud P-W (2019) Data-driven computation for history-dependent materials. *Comptes Rendus Mécanique* 347(11):831–844. <https://doi.org/10.1016/j.crme.2019.11.008>
  23. He X, He Q, Chen J-S (2021) Deep autoencoders for physics-constrained data-driven nonlinear materials modeling. *Comput Methods Appl Mech Eng* 385:114034. <https://doi.org/10.1016/j.cma.2021.114034>
  24. Capuano G, Rimoli JJ (2019) Smart finite elements: a novel machine learning application. *Comput Methods Appl Mech Eng* 345:363–381. <https://doi.org/10.1016/j.cma.2018.10.046>
  25. Tandale SB, Markert B, Stoffel M (2022) Smart stiffness computation of one-dimensional finite elements. *Mech Res Commun* 119:103817. <https://doi.org/10.1016/j.mechrescom.2021.103817>
  26. Tandale SB, Markert B, Stoffel M (2022) Intelligent stiffness computation for plate and beam structures by neural network enhanced finite element analysis. *Int J Numer Methods Eng*. <https://doi.org/10.1002/nme.6996>
  27. Koepe A, Bamer F, Markert B (2020) An intelligent nonlinear meta element for elastoplastic continua: deep learning using a new time-distributed residual u-net architecture. *Comput Methods Appl Mech Eng* 366:113088. <https://doi.org/10.1016/j.cma.2020.113088>
  28. Stoffel M, Gulakala R, Bamer F, Markert B (2020) Artificial neural networks in structural dynamics: a new modular radial basis function approach versus convolutional and feedforward topologies. *Comput Methods Appl Mech Eng* 364:112989. <https://doi.org/10.1016/j.cma.2020.112989>
  29. van den Oord A, Dieleman S, Zen H, Simonyan K, Vinyals O, Graves A, Kalchbrenner N, Senior AW, Kavukcuoglu K (2016) Wavenet: a generative model for raw audio. [arxiv:1609.03499](https://arxiv.org/abs/1609.03499)
  30. Bai S, Kolter JZ, Koltun V (2018) An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. <https://doi.org/10.48550/ARXIV.1803.01271>
  31. Yu F, Koltun V (2015) Multi-scale context aggregation by dilated convolutions. <https://doi.org/10.48550/ARXIV.1511.07122>
  32. Wang C, Xu L-Y, Fan J-S (2020) A general deep learning framework for history-dependent response prediction based on u-seq2seq model. *Comput Methods Appl Mech Eng* 372:113357. <https://doi.org/10.1016/j.cma.2020.113357>
  33. Sutskever I, Vinyals O, Le QV (2014) Sequence to sequence learning with neural networks
  34. Bahdanau D, Cho K, Bengio Y (2016) Neural machine translation by jointly learning to align and translate
  35. Hochreiter S, Schmidhuber J (1997) Long short-term memory. *Neural Comput* 9(8):1735–1780. <https://doi.org/10.1162/neco.1997.9.8.1735>
  36. Stoffel M (2007) Experimental validation of anisotropic ductile damage and failure of shock wave-loaded plates. *Eur J Mech A Solids* 26(4):592–610. <https://doi.org/10.1016/j.euromechsol.2006.12.002>
  37. Stoffel M (2004) Evolution of plastic zones in dynamically loaded plates using different elastic-viscoplastic laws. *Int J Solids Struct* 41(24):6813–6830. <https://doi.org/10.1016/j.ijsolstr.2004.05.060>
  38. Stoffel M, Schmidt R, Weichert D (2001) Shock wave-loaded plates. *Int J Solids Struct* 38(42):7659–7680. [https://doi.org/10.1016/S0020-7683\(01\)00038-5](https://doi.org/10.1016/S0020-7683(01)00038-5)
  39. Stoffel M (2006) A measurement technique for shock wave-loaded structures and its applications. *Exp Mech* 46(42):47–55. <https://doi.org/10.1007/s11340-006-5870-5>
  40. Stoffel M (2005) An experimental method to validate viscoplastic constitutive equations in the dynamic response of plates. *Mech Mater* 37(12):1210–1222. <https://doi.org/10.1016/j.mechmat.2005.06.001>
  41. Stoffel M, Schmidt R, Weichert D (1998) Vibrations of viscoplastic plates under impact load; [organised by Aristotle University of Thessaloniki, Greece; Wessex Institute of Technology, UK]. In: Jones N (ed) Structures under shock and impact V: fifth international conference, SUSI 98, held at Thessaloniki, Greece in June 1998. WIT transactions on the built environment, vol 35, pp 299–308. Computational Mechanics Publications, Southampton. <https://doi.org/10.2495/SU980271>
  42. Goodfellow I, Bengio Y, Courville A (2016) Deep learning. MIT Press
  43. Wang Z, Li C, Lin P, Rao M, Nie Y, Song W, Qiu Q, Li Y, Yan P, Strachan JP, Ge N, McDonald N, Wu Q, Hu M, Wu H, Williams RS, Xia Q, Yang JJ (2019) In situ training of feed-forward and recurrent convolutional memristor networks. *Nat Mach Intell* 1:434–442
  44. Rumelhart DE, Hinton GE, Williams RJ (1986) Learning representations by back-propagating errors. *Nature* 323:533–536. <https://doi.org/10.1038/323533a0>
  45. Gers FA, Schmidhuber J, Cummins F (2000) Learning to forget: continual prediction with LSTM. *Neural Comput* 12(10):2451–2471. <https://doi.org/10.1162/089976600300015015>
  46. Cho K, van Merriënboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, Bengio Y (2014) Learning phrase representations using RNN encoder–decoder for statistical machine translation
  47. Wu Y, Schuster M, Chen Z, Le QV, Norouzi M, Macherey W, Krikun M, Cao Y, Gao Q, Macherey K, Klingner J, Shah A, Johnson M, Liu X, Łukasz Kaiser Gouws S, Kato Y, Kudo T, Kazawa H, Stevens K, Kurian G, Patil N, Wang W, Young C, Smith J, Riesa J, Rudnick A, Vinyals O, Corrado G, Hughes M, Dean J (2016) Google’s neural machine translation system: bridging the gap between human and machine translation
  48. Luong M-T, Pham H, Manning CD (2015) Effective approaches to attention-based neural machine translation
  49. Jozefowicz R, Vinyals O, Schuster M, Shazeer N, Wu Y (2016) Exploring the limits of language modeling
  50. Zhang Z, Li M, Lin X, Wang Y, He F (2018) Multistep speed prediction on traffic networks: A graph convolutional sequence-to-sequence learning approach with attention mechanism. [arxiv:1810.10237](https://arxiv.org/abs/1810.10237)
  51. Khruikov V, Hrinchuk O, Mirvakhabova L, Oseledets IV (2019) Tensorized embedding layers for efficient model compression. *CoRR* [abs/1901.10787](https://arxiv.org/abs/1901.10787) <https://doi.org/10.10787arXiv:1901.10787>
  52. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser L, Polosukhin I (2017) Attention is all you need

53. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. *J Mach Learn Res* 15(56):1929–1958
54. Long J, Shelhamer E, Darrell T (2014) Fully convolutional networks for semantic segmentation. <https://doi.org/10.48550/ARXIV.1411.4038>
55. Remy P (2020) Temporal convolutional networks for Keras. GitHub
56. Kingma DP, Ba J (2017) Adam: a method for stochastic optimization
57. Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, Corrado GS, Davis A, Dean J, Devin M, Ghemawat S, Goodfellow I, Harp A, Irving G, Isard M, Jia Y, Jozefowicz R, Kaiser L, Kudlur M, Levenberg J, Mané D, Monga R, Moore S, Murray D, Olah C, Schuster M, Shlens J, Steiner B, Sutskever I, Talwar K, Tucker P, Vanhoucke V, Vasudevan V, Viégas F, Vinyals O, Warden P, Wattenberg M, Wicke M, Yu Y, Zheng X (2015) TensorFlow: large-scale machine learning on heterogeneous systems. Software available from <https://www.tensorflow.org/>
58. Chollet F et al. Keras. <https://github.com/fchollet/keras>
59. Li L, Jamieson K, DeSalvo G, Rostamizadeh A, Talwalkar A (2016) Efficient hyperparameter optimization and infinitely many armed bandits. [arxiv:1603.06560](https://arxiv.org/abs/1603.06560)
60. Bergstra J, Bengio Y (2012) Random search for hyper-parameter optimization. *J Mach Learn Res* 13:281–305
61. Chen CLP (1996) A rapid supervised learning neural network for function interpolation and approximation. *IEEE Trans Neural Netw* 7(5):1220–1230. <https://doi.org/10.1109/72.536316>

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