Polymer 261 (2022) 125395

Contents lists available at ScienceDirect

Polymer

journal homepage: www.elsevier.com/locate/polymer

Deep learning for predicting the thermomechanical behavior of shape memory polymers

Diego Segura Ibarra [a], Jacob Mathews [a], Fan Li [a], Hongfang Lu [c], Guoqiang Li [b], Jinyuan Chen [a,*]

[a] Department of Electrical Engineering, Louisiana Tech University, Ruston, LA, 71270, USA
[b] Department of Mechanical & Industrial Engineering, Louisiana State University, Baton Rouge, LA, 70803, USA
[c] China-Pakistan Belt and Road Joint Laboratory on Smart Disaster Prevention of Major Infrastructures, Southeast University, Nanjing, 210096, China

ARTICLE INFO

Keywords:
Shape memory polymer
Deep learning
Thermomechanical behavior

ABSTRACT

Thermomechanical constitutive modeling is essential for shape memory polymers (SMPs) to be used in engineering structures and devices. However, the classical method of deriving constitutive models is difficult, time consuming, and relies heavily on trial and error. In this work, we aim to decrease the time and resources needed to develop new thermomechanical models for SMPs. The method proposed in this work uses deep learning (DL) to predict the thermomechanical behavior of SMPs under thermomechanical cycles. Particularly, a semicrystalline two-way shape memory polymer (2W-SMP) is selected as an example. Predicting such behavior will give insight on the SMP properties and help validate its characteristics. In this paper, we have compared several DL models to find which one can predict the experimental thermomechanical behavior with the highest accuracy within a reasonable time frame. The results reveal that the fully connected neural network (FCNN) and the convolutional neural network (CNN) were the most accurate DL models. Overall, using one of the selected DL models, we can predict the results of new iterations of the experiment without spending as much time and resources.

1. Introduction

Since the early 2000s, shape memory materials have grown in popularity due to their wide range of applications. Out of the group of shape memory materials, shape memory polymers (SMPs) have stood out due to their low cost and remarkable shape memory effect [1]. SMPs are smart materials capable of returning from a deformed shape to their original shape in response to an external stimulus like change in temperature, pH, electric field, or magnetic field [2]. Due to the various stimuli, they can be applied in many fields like medicine [3], engineering [4], or textiles [5]. Despite the success behind SMPs, there is still a need to design new SMPs with better properties so that their application can be extended to more areas. However, discovering new SMPs with the desired properties, is a difficult and time-consuming process that relies on trial and error [6]. The current method to discover SMPs involves creating and synthesizing new SMPs by a bottom-up process and validating them through experimentation. Synthesizing the SMPs requires a lot of time and expertise, and validating them through experimentation is laborious and time consuming [6]. In this work, we aim to assist in the polymer discovery process by reducing the time needed to validate newly designed SMPs.

Validating newly designed SMPs needs researchers to verify if the polymers have the desired characteristics. Some of the important properties that are found through validation include the recoverable strain and stress levels of the polymer. These two parameters show how well the polymer can recover after deformation, and if it can recover while applying an external load to the material. A technique used to evaluate these properties is thermomechanical analysis (TMA). In TMA, the change in stress and strain of a material is measured while an external load is applied, and temperature is varied over time. This is the thermomechanical behavior. With this technique, the change in strain in the polymer is measured in response to time, temperature, and external load. Evaluating this behavior is important because it demonstrates how the polymer will behave under conditions that can appear in real world scenarios and help determine whether a polymer is suitable for a certain application. Additionally, by performing TMA over a variety of constraints, the properties of the polymer can be better understood. However, it can be time-consuming and resource intensive to perform TMA multiple times.

Modeling and simulation methods can be used to see how the polymer will respond to different conditions and estimate the change

* Corresponding author.
E-mail address: jinyuan@latech.edu (J. Chen).

https://doi.org/10.1016/j.polymer.2022.125395
Received 22 July 2022; Received in revised form 24 September 2022; Accepted 27 September 2022
Available online 10 October 2022

in strain of a SMP without performing experimental TMA. Using simulations, laborious experiments can be avoided, and the time and resources needed to validate SMPs can decrease. In this work, we take the advantage of deep learning (DL) to perform the simulations. DL is an emerging method that can use large amounts of data to make fast and accurate predictions. There are a variety of DL models that could be used to conduct the simulation. In this paper, we will compare eight neural network models to predict the thermomechanical behavior of SMPs.

Machine learning (ML) is a superset of DL that has been proven useful for many material science applications [7,8]. ML has gained popularity in this field because it is less computationally intensive and faster than current material simulation methods like molecular dynamics (MD) and density functional theory (DFT) [6]. Some of the notable applications of ML to material science include predicting properties and discovering novel materials [8]. For instance, Lee et al. showed that by utilizing inverse ML strategies, they can design new metal alloys with desired yield strength and ultimate tensile strength [9]. Apart from metal alloys, ML has also been applied in areas such as, nanomaterials [8], polymers [6,10–14], and drugs [8]. In the area of polymer science, ML has had a great impact on the design of novel polymers [6,10–14]. For example, ML has been used to design polymers with high thermal conductivity [11], large energy band gap [15], desired glass transition temperatures [6,11–14], or a desired dielectric constant [15]. Only a few of the works that we have investigated have focused on the application of ML to SMPs [6,14]. The works that focus on ML for SMPs look for ML models that can design new SMPs based on a desired recovery stress and glass transition temperature [6,14]. An example of such works is seen through Yan et al. who discovered new thermoset shape memory polymers (TSMPs) with high recovery stress and moderate glass transition temperature by using novel ML algorithms [14].

To the best of our knowledge, the applications of DL for SMPs have focused on discovering new SMPs, but there is a lack of research dedicated towards using ML to predict the thermomechanical behavior of SMPs. The thermomechanical behavior of the polymer is observed through TMA. Predicting this behavior is important because it allows researchers to see how the polymer will behave over time under external load and temperature change, which helps them validate newly designed polymers. Actually, thermomechanical behavior or thermomechanical constitutive law is not only critical for validating newly designed SMPs, it is also essential for designing load carrying structures and devices made of SMPs. Despite the lack of works focused on ML methods to predict the thermomechanical behavior, there is extensive research dedicated towards creating mathematical models to predict this behavior [16–25]. For instance, Heuchel et al. [24] used a modified Maxwell–Weichert model to describe the stress relaxation of radiopaque polyether urethane. Also, Lu and Huang [25] were able to use the Vogel–Fulcher–Tammann thermodynamic framework to simulate the temperature dependent relaxation behavior. However, most of the mathematical models consist of a large number of model parameters, which need to be determined through curve-fitting experimental results. Usually, this curve-fitting process is very time consuming, and most of the time, the model can only be applied to specific SMPs on which the curve-fitting is based. Furthermore, although the discrepancy between model prediction and experimental observation for most mathematical model is acceptable in engineering, there is a large room for improvement.

In recent years, machine learning (ML) has been used to establish constitutive laws for engineering materials such as solid [26] for instance rocks [27], and liquids [28]. For example, Furukawa and Yagawa [29] and Ghaboussi et al. [30] used supervised learning, and trained feed-forward neural networks to obtain constitutive laws. Kirchdoerfer and Ortiz [30] used supervised learning process and constrained optimization approach to obtain constitutive models. Their strategy was to minimize the discrepancy between measured and predicted responses. They also used several physical laws such as conservation of mass, conservation of energy, conservation of momentum, and thermodynamics law as constraints in the minimization problems. Ibanez et al. [31] has used this method in plasticity problems. They introduced the yield surface and enforced perfect plasticity. To our knowledge, however, there are no ML assisted thermomechanical constitutive models for SMPs. In this work, we aim to fill in the gap by applying ML strategies to model the thermomechanical behavior of SMPs. Specifically, we are interested in using DL to simulate how the polymer will respond to thermomechanical cycles, *i.e.,* varying temperatures and applied stresses over time.

In our work, we have chosen several neural network approaches to predict the change in strain of SMPs. The selection of DL models in this paper is based on neural networks that have found success for solving time series forecasting problems. In this paper, the objective is to predict the change in strain of an SMP as a function of time, temperature, and external load. Since the variable to be predicted is a time dependent variable, DL models for time series forecasting are suitable to make the prediction. Each of the selected models has their own advantages and disadvantages, and through this work we aim to see which neural network can predict the thermomechanical behavior of SMPs with the best prediction accuracy.

Overall, the contributions of this paper are the following:

- Demonstrate the capabilities of DL to predict the thermomechanical behavior of SMPs.
- Recognize the best performing DL model to predict the thermomechanical behavior of a SMP out of the selected DL models.
- Propose a general scheme to predict the thermomechanical behavior of SMPs.

The rest of this paper is organized as follows: Section 2 discusses methodology used in this work, which includes an explanation of the data, preprocessing steps, DL models, and error evaluation metrics; Section 3 displays the performance of each DL model; Section 4 discusses and compares the results outlined in Section 3; Section 5 summarizes our findings and outlines the future direction of this work.

2. Methodology

To predict the change in strain in a SMP, we developed a simple DL scheme that uses the data from experiments to model the thermomechanical behavior of a SMP. An overview of the entire scheme is shown in Figs. 1 and 2.

The scheme goes as follows: first, the TMA data is gathered from experimentation on a certain SMP. This data is then organized into multiple datasets where each one represents a TMA experiment under different constraints (see Section 2.1). Second, each dataset is formatted to prepare it for each DL model (see Section 2.2). After each dataset is preprocessed, they are concatenated into a single group. Third, the preprocessed data is inputted to the DL model for training (see Section 2.3). Once the DL model is trained, it can be used to predict the change in strain of the polymer by inputting a dataset containing the desired constraints at which the polymer would be evaluated. The performance of the DL model is then calculated by comparing the predicted behavior with the actual behavior using two error metrics (see Section 2.3.3). Figs. 1 and 2 show the training and prediction processes, respectively.

In this paper, two datasets containing the experimental TMA of a certain SMP will be used to test the DL models. These two datasets will be used in two ways. First, the DL models will be trained with a section of one dataset and they will be tasked to predict the rest of that dataset. Second, the DL models will be trained with one complete dataset and they will be tasked to predict the second dataset. Both these test will determine if the DL model can understand the relationships within the data and demonstrate how they can be applied.

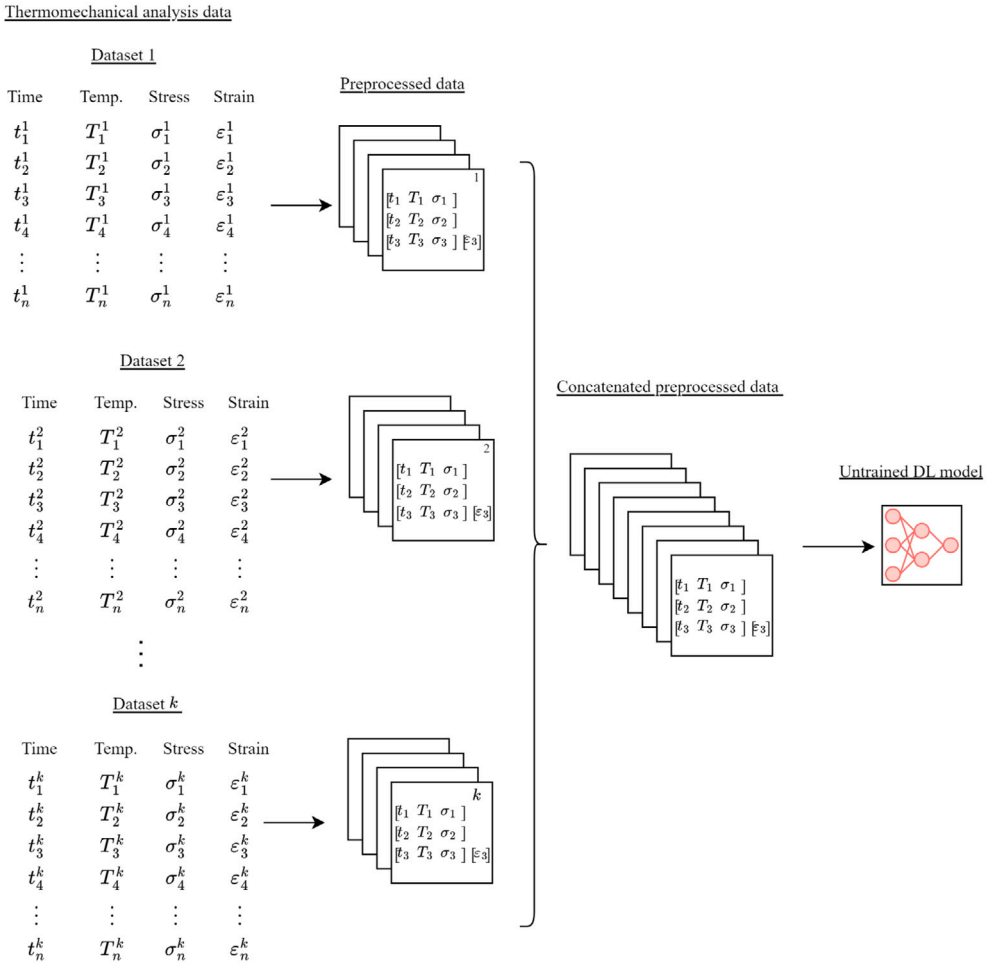

Fig. 1. Training process of DL model.

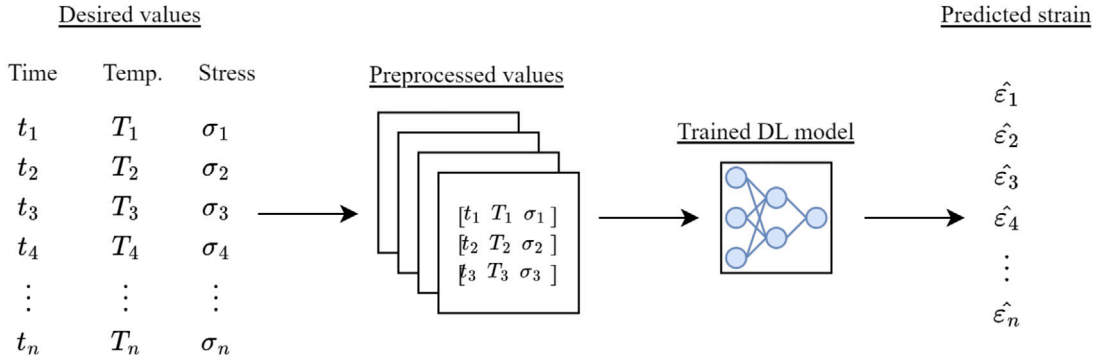

Fig. 2. Prediction process to find the change in strain.

2.1. Data

To train and validate our selected DL models, we use data obtained from TMA of a two-way shape memory polymer (2W-SMP) known as crosslinked *cis* polybutadiene (cPBD) [32]. A 2W-SMP is a type of SMP that shifts reversibly between its permanent shape and temporary shape as temperature cycles. As compared to one-way shape memory polymers (1W-SMPs), which usually use glass/vitrification transition or crystallization/melting transition to control the shape memory effect, the thermomechanical behavior of 2W-SMP is more complex. For the cPBD, which is a semicrystalline 2W-SMP, its thermomechanical behavior is controlled by both rubber elasticity at rubbery state, and melt/crystallization transition at temperature below the crystallization temperature. Also, although several thermomechanical constitutive models have been developed over the years [16,24,25,33–36], the discrepancy between the model prediction and experimental observation is still considerable. Therefore, to demonstrate the capability of ML, we choose this more complex system in this study.

The data obtained from the TMA of cPBD is divided into two datasets containing three time dependent variables: stress, temperature, and strain. Stress and temperature are controlled variables which are determined before performing TMA. Strain is a variable that depends on the changes of stress and temperature over time. As the controlled variables are varied over time, a strain curve is created that displays

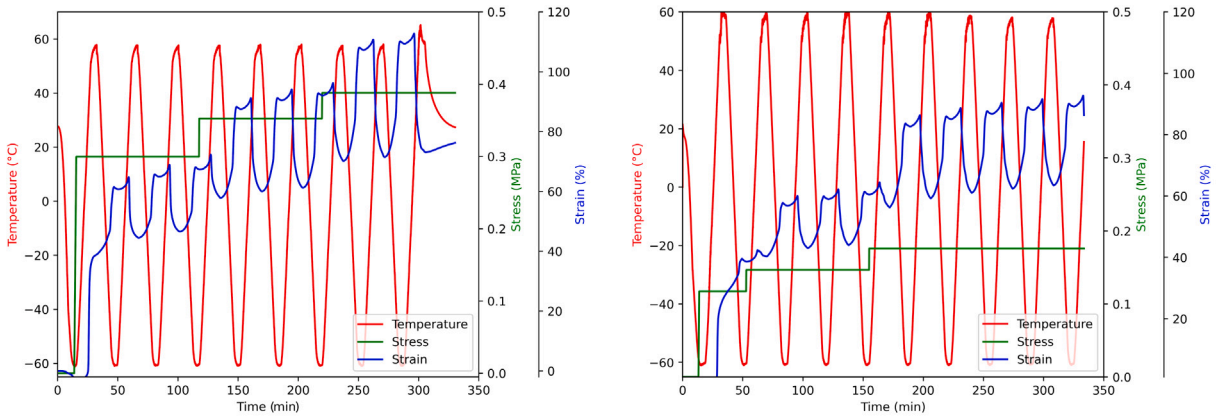

Fig. 3. Plots of the two datasets used in this work: (a) dataset 1 and (b) dataset 2.

Table 1
Statistical information of dataset 1 and dataset 2.

	Dataset 1			
	Minimum	Maximum	Mean	Standard deviation
Temperature (°C)	−61.060	62.589	−3.795	42.312
Stress (MPa)	9.770×10^{-4}	0.176	0.153	0.037
Strain (%)	−2.116	92.757	57.278	24.695
	Dataset 2			
	Minimum	Maximum	Mean	Standard deviation
Temperature (°C)	−61.168	65.185	0.975	41.961
Stress (MPa)	8.820×10^{-5}	0.388	0.332	0.079
Strain (%)	−2.610	112.821	66.207	26.701

the changes in deformation of the 2W-SMP. In each dataset, there are two separate TMAs for cPBD, where the temperature is similar, but the stress is varied over different ranges. The oscillating temperature is dependent on the melting and crystallization transition temperatures. The maximum temperature is above the melting temperature and the minimum is below the crystallization temperature. Since in both datasets the TMA of the same polymer is evaluated, the temperature oscillation is similar. On the other hand, in the first and second dataset, the stress varies from approximately 0.3 MPa to 0.4 MPa and from 0.1 MPa to 0.2 MPa, respectively. The different external loads applied in each dataset lead to distinct strain curves. Within the two datasets, there are a total of 38,802 datapoints, from which 19,820 correspond to dataset 1, and 18,982 correspond to dataset 2. A visual representation of these datasets is shown in Figs. 3(a) and 3(b). Information on the statistical characteristics of that data can be found in Table 1.

2.2. Data preprocessing

In this section, an explanation of the steps taken to prepare the data for each DL model is presented.

2.2.1. Normalization

In ML/DL problems the scale of data has the potential to unintentionally bias the prediction [37]. Table 1 shows that the temperature and stress do not have consistent averages or standard deviations. To ensure that our models give equal weight to each variable, each input variable was normalized as follows:

$$z_i = \frac{x_i - \mu}{\sigma} \tag{1}$$

where z_i is the new normalized value, x_i is the original value, μ is the average of all values, and σ is the standard deviation. The values of μ and σ can be found using the following equations

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{2}$$

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \mu)^2}{N}} \tag{3}$$

where N is the total number of data points.

2.2.2. Sequencing

In order for DL models to interpret the data properly, the data must first be split into inputs and outputs, labeled as X and y, respectively. As an input, the controlled variables are used (*i.e.,* time, temperature, and stress), and as an output, the dependent variable is used (*i.e.,* strain). Before passing the data into each DL model, the input values must be further organized. To predict a strain value at a given instant, providing the time, temperature, and stress values at that moment would be insufficient. In addition to the time, temperature, and stress values at that moment, the historical values of time, temperature, and stress that led up to that moment must also be provided to make an accurate prediction. This is because the strain of an SMP measured through TMA is dependent on both the current and previous values of the controlled variables. For this reason, the input values must be organized in such a way that they can provide the needed context to a certain output value. In our work, the input data was organized into sequences containing multiple instances of time, temperature, and stress values. The output value of these sequences is the strain value that belongs to the last instant in the input sequence. An example of the sequencing process is described in Fig. 4.

The number of instances present in each sequence is labeled as n. It is important to note that splitting the data as such, also removes some of the initial output values. For example, if three instances are used (like in Fig. 4), then the first input will have the first three instances of time, temperature, and stress values, and the output would be the strain value corresponding to the third instance of time, temperature, and stress. In this example, the first and second strain values are lost. Therefore, choosing a large n runs the risk of forgetting many initial strain values.

This data format informs each output value with a number of input data points determined by n. Additionally, each DL model used a different n. The quantity of n used in each DL model is summarized in Tables 2 and 3.

Table 2
RMSE and MAPE of the predictions of each DL model in datasets 1 and 2. This table includes the computation time for each DL model and prediction and the number of instances, n, used to format the data for each DL model. All models were trained with an Intel(R) Xeon(R) Silver 4210 processor.

Learning models	n	Dataset 1			Dataset 2		
		RMSE	MAPE (%)	Time (s)	RMSE	MAPE (%)	Time (s)
FCNN	1200	5.168	4.270	270	3.003	3.458	251
CNN	1000	2.755	2.149	1345	1.462	1.671	750
LSTM	100	3.609	3.490	1300	1.680	1.717	1150
BiLSTM	100	2.457	2.580	1300	1.974	2.186	850
CNN-LSTM	400	3.857	4.247	800	3.727	3.721	700
CNN-BiLSTM	400	4.051	3.965	850	1.099	1.036	1108
ConvLSTM	81	6.523	7.246	1450	4.962	4.490	1400
Ensemble	N/A	2.467	2.403	2895	1.042	1.100	2151

Table 3
RMSE and MAPE of the predictions made by FCNN, CNN, and LSTM.

	n	RMSE	MAPE (%)
FCNN	4000	6.371	7.330
CNN	1000	8.632	11.739
CNN-LSTM	400	8.9	12.372

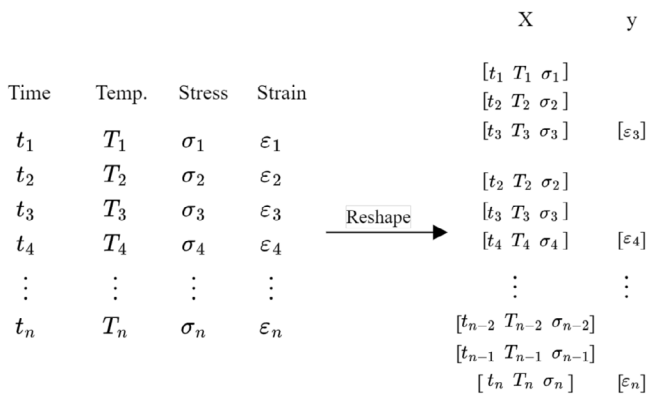

Fig. 4. Splitting and sequencing the input data.

2.2.3. Dimensional changes

The last step to preprocess the data was changing their dimensions. Every neural network model accepted input data in different dimension. For instance, the convolutional neural network (CNN) expects input data in three dimensions while convolutional long short-term memory (ConvLSTM) neural network expects data in five dimensions. After the sequencing step, data was presented in three dimensions, which is adequate for models like the fully connected neural network (FCNN), CNN, long short-term memory (LSTM), bidirectional LSTM (BiLSTM), and ensemble. However, for models like CNN-LSTM, CNN-BiLSTM, and ConvLSTM, the data had to be reshaped to fit the required input shape. For CNN-LSTM and CNN-BiLSTM four dimensions are needed and for ConvLSTM five dimensions are needed. Once the dimensional changes are finished, the data is ready to be used by the corresponding DL model.

2.3. Deep learning models

To predict the strain of a 2W-SMP, we gathered a selection of neural networks that have found success for solving time series forecasting problems. The DL models used are the FCNN, CNN, LSTM, BiLSTM, CNN-LSTM, CNN-BiLSTM, ConvLSTM, and an ensemble neural network. We organized the selection of DL models into two categories: core and hybrid. Core models are common neural networks that have been proven successful for problems with time dependent data, while hybrid models have the potential for success due to their similarity with the aforementioned models. Each neural network will be tasked to predict how the strain of a 2W-SMP will respond to changes in time, temperature, and stress. To this end, the models will take a series of desired time, temperature, and stress values as inputs and output the corresponding changes in shape, *i.e.,* strain. A brief explanation of the DL models can be found on the appendix.

2.3.1. Core models

Three DL models that have seen repeated success for solving time series forecasting problems are: FCNN, CNN, and LSTM [38]. These models were chosen due to their capability of finding non-linear relationships and their popularity for solving time series forecasting problems. Each of these models have special characteristics that can be useful for predicting the behavior of 2W-SMPs. A visual representation of these models with specific hyperparameter information is shown in Fig. 5.

2.3.2. Hybrid models

In addition to the core models, we implemented five other models based on similar principles to FCNN, CNN, and LSTM. The purpose of using these other models is to experiment with neural networks that either expand or combine the characteristics of the FCNN, CNN, and LSTM. With the hybrid models, we can have a larger scope on neural networks and have a better idea of which models work best to solve our problem. A visual representation of these models with their hyperparameters is shown in Fig. 6.

2.3.3. Error metrics

After developing and training each model, we evaluated their performance by predicting the strain curves from datasets 1 and 2 (shown in Figs. 3(a) and 3(b)) using only the corresponding time, temperature, and stress values. We compared the predicted and experimental results and used two error metrics to evaluate their error. The two error metrics used were the Root Mean Squared Error (RMSE) and Mean Absolute Percent Error (MAPE). These results are shown in Figs. 7, 8, and 9 and the error from each model is detailed in Tables 2 and 3.

In this paper, we use Eqs. (4) and (5) to calculate RMSE and MAPE

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{i=1}^{N}(A_i - F_i)^2}{N}} \tag{4}$$

$$\mathrm{MAPE} = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{A_i - F_i}{A_i}\right| \times 100\% \tag{5}$$

where A_i is the actual value, F_i is the forecasted value, and N is the number of data points.

3. Results

After the data was preprocessed, it was inputted to the DL models to make predictions. First, the DL models were trained with each dataset individually. For the case of dataset 1, the data was split using an 8:2 ratio. The DL models were trained with 80% of the data and tested

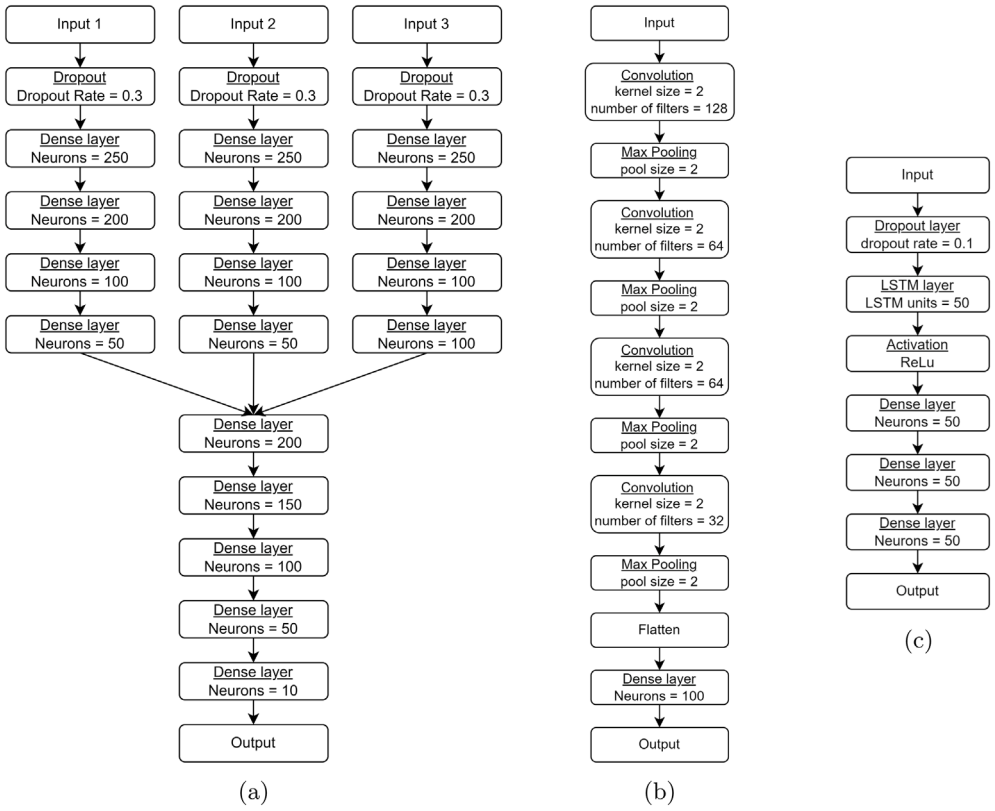

Fig. 5. Visual representation of core neural network models where (a) shows the FCNN model, (b) shows the CNN model, and (c) shows the LSTM model.

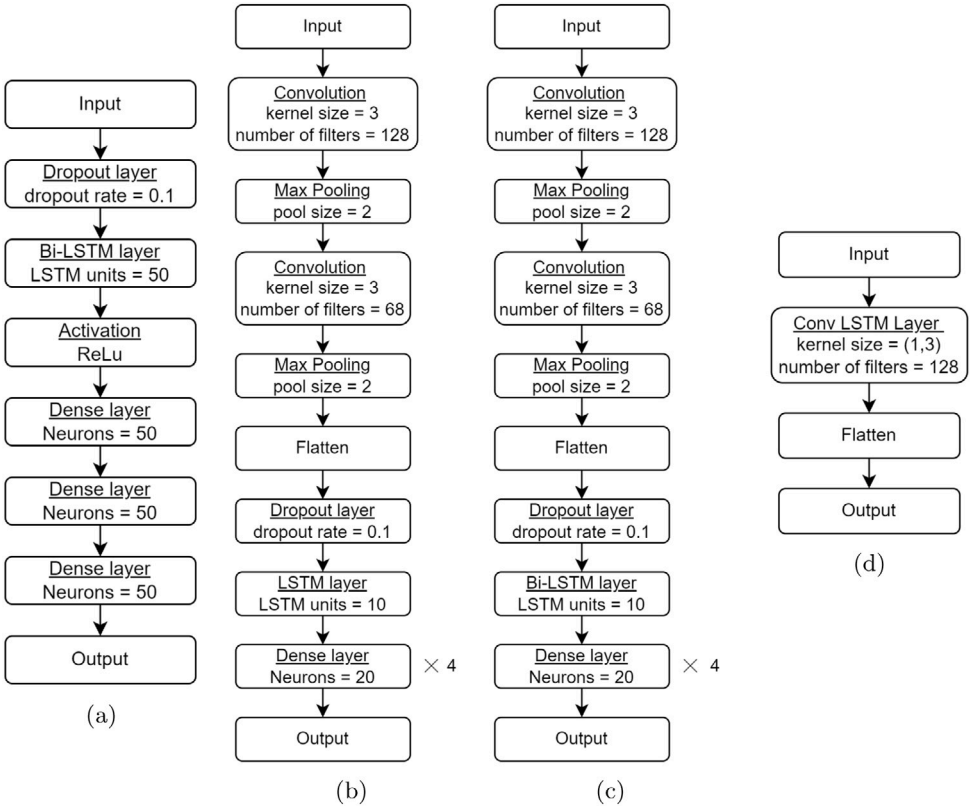

Fig. 6. Visual representation of hybrid neural network models where (a) shows the BiLSTM model, (b) shows the CNN-LSTM model, (c) shows the CNN-BiLSTM model, and (d) shows the ConvLSTM model.

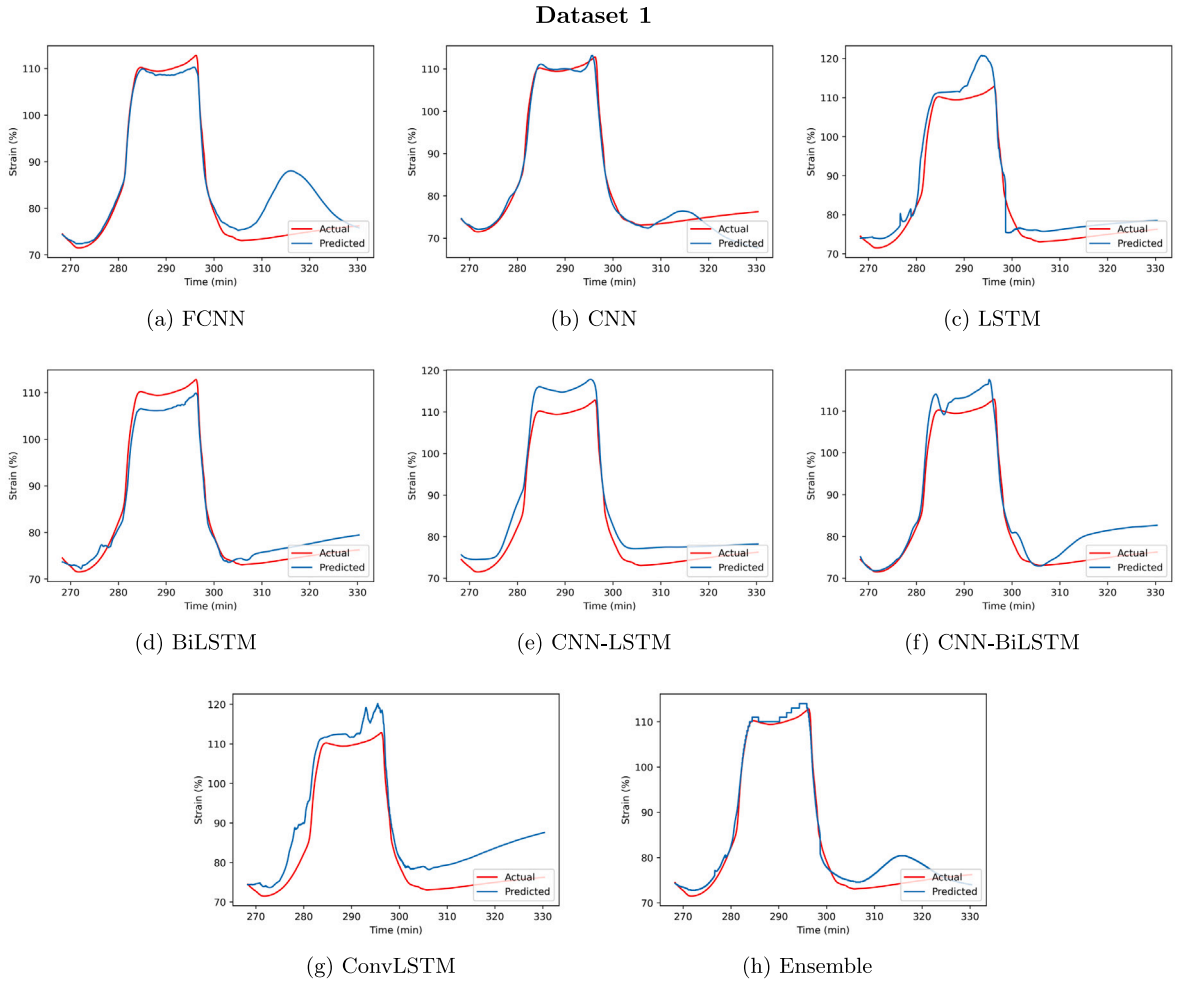

Fig. 7. Prediction performance for dataset 1 by the following DL models: (a) FCNN, (b) CNN, (c) LSTM, (d) BiLSTM, (e)CNN-LSTM, (f) CNN-BiLSTM, (g) ConvLSTM, and (h) Ensemble.

with the remaining 20%. The same occurred for dataset 2, but instead of using an 8:2 ratio, a 7:3 ratio was used. Figs. 7 and 8 show the performance of the core and hybrid models for predicting the strain in datasets 1 and 2, respectively. Table 2 compares the errors from the forecast made by each DL model for datasets 1 and 2 and displays the training time for all models. Second, the DL models were trained with the entirety of dataset 1 and were tasked to predict dataset 2. For this case, only the models with a better performance are provided. These results are shown in Fig. 9 and the error is displayed in Table 3. The testing time was not recorded due to the little time the DL models needed to make a prediction.

Let us first look at the prediction performance of each deep learning model for predicting the strain in dataset 1, which can be found in Fig. 7. Focusing on the core models, it can be observed that CNN had the lowest prediction error in comparison to LSTM and FCNN with an RMSE of 2.755 and an MAPE of 2.149% (see Table 2). FCNN performed the worst with error values of 5.168 and 4.270% for RMSE and MAPE, respectively. LSTM was between FCNN and CNN with an RMSE of 3.609 and an MAPE of 3.490%. From the hybrid models, the BiLSTM and ensemble neural networks had the lowest error compared to the rest of the models in the hybrid section. The worst performing model in this section was the ConvLSTM model as Table 2 shows that its error was the highest. After the ConvLSTM model, CNN-BiLSTM, CNN-LSTM, BiLSTM, and ensemble follow in order of highest to lowest error, with errors ranging from 2.457 to 6.523 and 2.403% to 7.246% for RMSE and MAPE, respectively.

Fig. 8 shows the prediction performance of the learning models when predicting the strain from dataset 2. For the core models, the prediction made by the CNN model had the least error in comparison to FCNN and LSTM, which is similar to the prediction of dataset 1. For the hybrid models, it can be seen that the prediction made by the CNN-BiLSTM and ensemble had the least error. For dataset 2, the order from worst performing to best performing of the hybrid models is ConvLSTM, CNN-LSTM, BiLSTM, ensemble, and CNN-BiLSTM. Overall, it can be pointed out that CNN and Ensemble produced the lower errors more consistently for the core and hybrid models, respectively. Between the two models, CNN produced a lower error for the MAPE in the prediction of dataset 1 and dataset 2. Meanwhile, Ensemble surpassed CNN in the RMSE for the prediction of dataset 1 and dataset 2. However, while ensemble's prediction is on the same level as CNN's prediction, ensemble takes significantly longer time than CNN. It is worth noting that Yan et al. [16] modeled the same cPBD system using mathematical model. It is clear that the ML prediction is more accurate than the mathematical thermomechanical model.

Now, let us take a look at the performance of the DL models tasked to predict dataset 2 after being trained with dataset 1. In this case, only the three best performing DL models are shown. Out of the tested DL models, FCNN, CNN, and CNN-LSTM gave the lowest error. The results are shown in Fig. 9 and the error in Table 3. Table 3 also displays the quantity of n needed to make such predictions. In this test, FCNN was found to make the most accurate prediction with an RMSE of 6.371 and an MAPE of 7.330%. CNN and CNN-LSTM follow FCNN with the error from CNN being lower than that of CNN-LSTM.

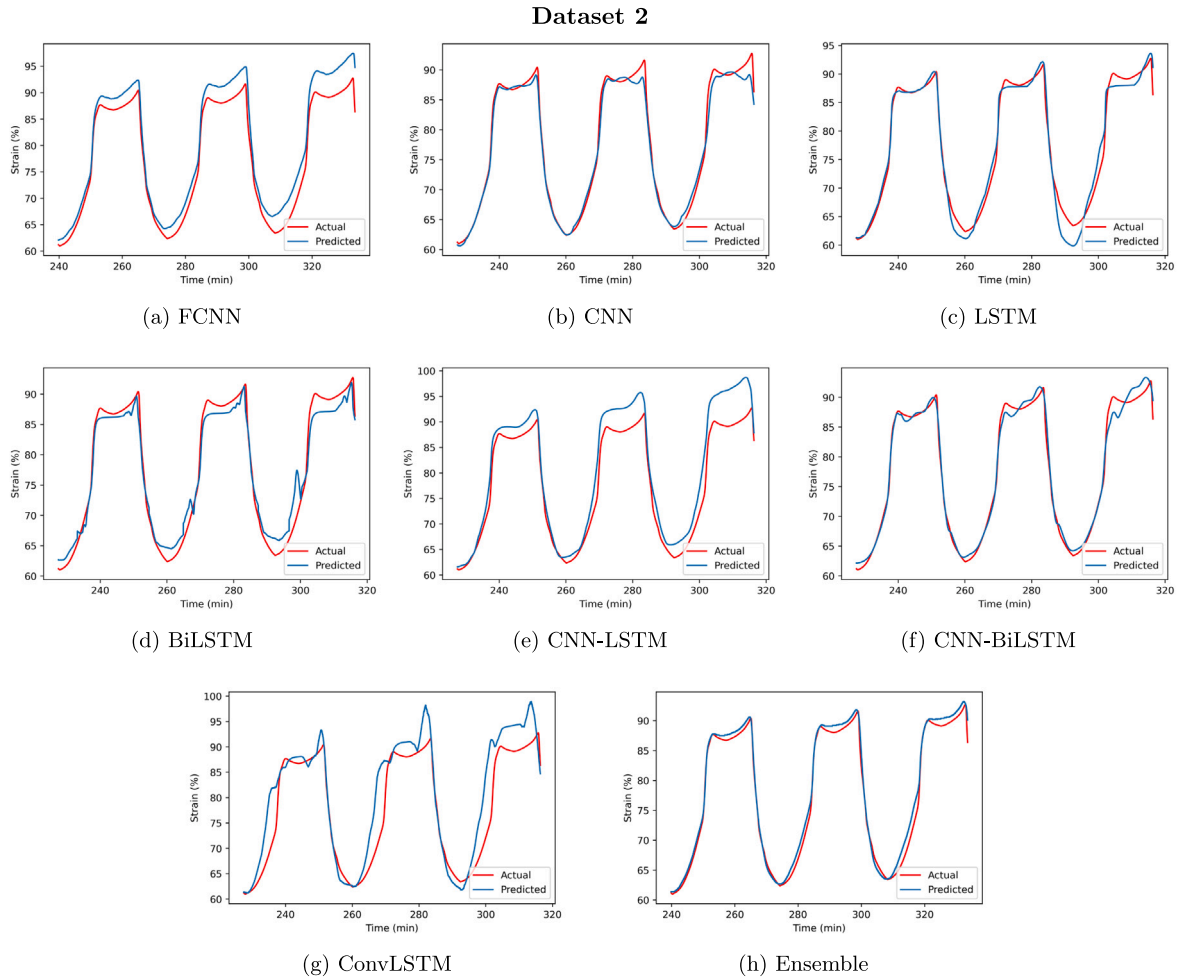

Fig. 8. Prediction performance for dataset 2 by the following DL models: (a) FCNN, (b) CNN, (c) LSTM, (d) BiLSTM, (e)CNN-LSTM, (f) CNN-BiLSTM, (g) ConvLSTM, and (h) Ensemble.

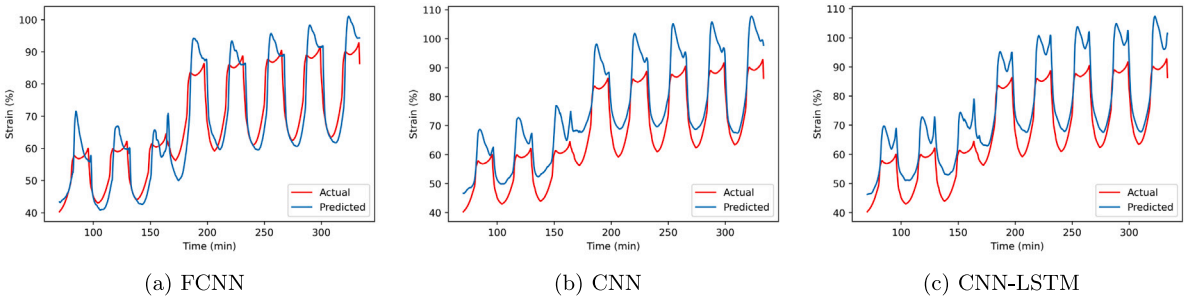

Fig. 9. Prediction performance for dataset 2 by the following DL models: (a) FCNN, (b) CNN, and (c) CNN-LSTM.

4. Discussion

First, the predictions shown in Figs. 7 and 8 will be evaluated. In Section 3, it was determined that CNN and ensemble produced a lower error in comparison to the other DL models. A possible reason why CNN performed well is due to the large quantity of *n*. Recall from Section 2.2.2, the quantity of *n* determines the amount of context the neural network will use to determine a certain strain value. This context is made by using multiple instances of time, temperature, and stress values that lead up to the strain value that will be predicted. CNN used 1000 instances of time, temperature, and stress values to calculate each singular strain value. The large *n* accompanied with CNN's ability to extract important information from the data, contributed to CNN's low prediction error. A large *n* is also used in FCNN; however, its error was not as low as that of CNN. This likely occurred because FCNN lacks the complexity and capabilities of CNN. Still, the prediction made by FCNN is comparable with the other DL models. Out of the DL models that produced a low prediction error, LSTM and BiLSTM are shown to make predictions with low error without needing a large *n*. Just using 100 instances, LSTM and BiLSTM were able to make one some of lower prediction errors in comparison to the rest of the DL models. A drawback observed from CNN, LSTM, and the hybrid models is a long training time. While complex models like CNN, LSTM, and the hybrid models were able to make good predictions, they took more time to train. It should also be mentioned that the quantity of *n* also influences the length of the training time but not as much as the complexity of

the DL model. The ensemble neural network is a special case because it does not use an exact value for n, rather it simply averages the individual results of FCNN, CNN, and LSTM. In specific, ensemble computes the predictions for FCNN, CNN, and LSTM separately and averages the predictions made by each neural network. As a result, ensemble has the highest training times by far. Still, ensemble provided one of the lowest prediction errors. The RMSE made by the prediction from ensemble for datasets 1 and 2 is lower than all the other models. Overall, the DL models used in this work are shown to be capable of learning the TMA data in one dataset well enough to predict future values in that TMA experiment. Out of the tested DL models, CNN and ensemble are shown to make predictions with consistent low error values. However, since ensemble takes a considerable amount of time to use, CNN would be the preferable model.

At this point, it can be concluded that the DL models can learn the relationship between strain and time, temperature, and stress within one TMA dataset and are able to predict future values of that TMA. To further determine the capabilities of the DL models, they were tasked to predict a completely unseen experimental TMA. This test would help determine whether these DL models could predict new versions of TMA for a certain polymer. To carry out this test, the DL models were trained with only dataset 1. Then, they were tasked to predict the strain curve from dataset 2, using the time, temperature, and stress in that dataset as inputs. The results shown in Fig. 9 and Table 3 reveal that FCNN, CNN, and CNN-LSTM are able to predict the unseen data with high accuracy, in terms of RMSE performance. From these models, FCNN performed the best with the lowest RMSE and MAPE. The good performance of FCNN shows that the DL model was able to understand how strain is related to time, temperature, and stress in a certain SMP. It can be noted that while the prediction from these DL models is not perfect, it is good enough to give an idea of the strain levels of the polymer at a certain scenario of time, temperature, and stress. In practice, it can also be beneficial to use FCNN due to its low training time.

After having determined the better performing DL model for this problem, it can now be applied. In essence, the complete scheme or framework can be used to determine the thermomechanical response of SMPs. To use the scheme, one would give it the conditions at which one seeks to evaluate the SMP. That is, the scheme will receive information on the temperatures to oscillate, the desired stress to be applied to the SMP, and the amount of time to run the experiment. With this information, the scheme will output a strain curve which will give insight in how the deformation of the SMP will change according to those conditions. With this data, the recovery strain at the given stress is observed. This observation can help the investigator verify the characteristics of the SMP and determine whether the SMP is suitable for a certain application. With this scheme, the amount of time and resources needed to test SMPs can be heavily reduced. In addition to SMPs, this scheme can also be used to evaluate the thermomechanical response of other materials. To use it in other materials, the data on the thermomechanical response of the material in question needs to be provided to the model. Overall, through this work we are able to demonstrate that DL models can be used as a method to simulate the thermomechanical behavior of the SMP. However, the current work also has some limitations. The main limitation is that the DL model developed in this paper can only predict the thermomechanical response of one material at a time. In this paper, a method to predict the thermomechanical behavior of cPBD is shown. If the thermomechanical behavior of another material is desired, then the DL model would have to be retrained with data from the thermomechanical behavior of the material in question. To overcome this limitation, we aim to generalize this model further by adding the chemical structure of the polymer as an input to the model. The addition of this input parameter would allow the DL model to predict the thermomechanical behavior of various polymers at a time, which would expand the application of this work.

5. Conclusion and future works

In this paper, we designed multiple DL models that were tasked with predicting how the strain curve would change in response to changes in time, temperature, and stress. Predicting the strain curve allows us to see the shape memory behavior of a SMP that is important in the validation process. To evaluate each model, we predicted the strain curve of a known 2W-SMP subjected to thermomechanical cycles under varying loads. By using the time, temperature, and stress values used in the thermomechanical cycles, we predicted the strain curve of each dataset that was shown in Fig. 3. The results show that the models predicted the strain curve in both datasets with high accuracy and reveal that DL algorithms can predict the shape memory behavior of a SMP with very low error. When comparing the DL models in terms of error, it becomes clear that the FCNN, CNN, and ensemble models have better performance. The CNN and ensemble models are shown to have lower error values when trained with a section of the data and tasked to predict the rest. Additionally, FCNN was able to make a prediction with high accuracy on an completely unseen TMA experiment. The good performance from these neural networks reveals that they can fit this data better than the rest. To apply this model, one can input the values at which the temperature would oscillate, the load that would be applied, and the amount of time the experiment would be run for. From these inputs, the model will output the thermomechanical behavior of the SMP. By using this model, we can evaluate polymers quickly with less experimentation, which can speed up the polymer design process.

In the future, we will include the polymer structure as an input parameter so that our models can be used for both SMP discovery and validation. To accomplish this goal, we would need to explore more polymer structures and acquire more experimental data of SMPs. We would also need to restructure the neural network to include the polymer's chemical structure using chemical informatics. With enough data, the model will be able to correlate the structure of the SMP with its thermomechanical response. Notably, the glass transition temperature of a polymer is needed to select the proper input temperatures for TMA. Therefore, if we can find the glass transition temperature of a SMP before it is synthesized and add polymer structure to the models, we can predict TMA of a SMP that has not been synthesized.

CRediT authorship contribution statement

Diego Segura Ibarra: Developed the deep learning models, Wrote and revised the manuscript. **Jacob Mathews:** Developed the deep learning models, Wrote and revised the manuscript. **Fan Li:** Developed the deep learning models, Wrote and revised the manuscript. **Hongfang Lu:** Developed the deep learning models, Wrote and revised the manuscript. **Guoqiang Li:** Provided the polymer data, Revised the manuscript. **Jinyuan Chen:** Conceptualized the study, Revised the manuscript, Supervised the study.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This work was supported by the National Science Foundation (NSF), USA Established Program to Stimulate Competitive Research (EPSCoR)-Louisiana Materials Design Alliance (LAMDA) program under Grant OIA-1946231.

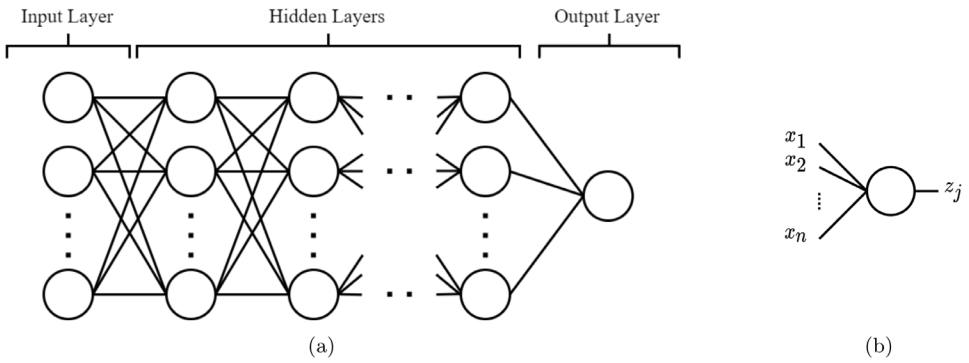

Fig. 10. Visual representation of a general FCNN where (a) shows the complete FCNN model and (b) shows a section of the FCNN model displaying a single neuron.

Appendix

A.1. Deep learning models

In this work, eight different types of neural networks were used. The types of neural networks used were the fully connected neural network (FCNN), convolutional neural network (CNN), long short-term memory (LSTM), bidirectional LSTM (BiLSTM), CNN-LSTM, CNN-BiLSTM, ConvLSTM, and ensemble. To optimize the neural networks, a trial-and-error approach was used to find the best combination of hyperparameters from a selection of viable hyperparameter values. Some hyperparameter values are consistent for the all of the selected models. Specifically, all of the neural networks use Adam as the optimizer and the mean squared error as the loss function. Other hyperparameters, like the number of neurons or hidden layers, differ for each DL model.

A.1.1. Fully connected neural network

A FCNN is a specialized feedforward neural network that consists of a fully connected input layer, a fully connected output layer, and one or more fully connected hidden layers [39,40]. In an FCNN, as with the rest of the neural networks, the objective is to map a set of inputs to an output. The FCNN achieves that objective by using a series of interconnected hidden layers where each neuron within each layer is connected to all of the neurons in the following layer, hence fully connected. These hidden layers are responsible for manipulating the input data using weights, biases, and activation functions to generate an output. Fig. 10 shows a visual representation of a general FCNN, where the output of a single neuron in the network is given by

$$z_j = \mathrm{g}\left(\sum_{i=1}^{n} w_{ij} x_i + b_j\right) \tag{6}$$

where the z_j is the output of the jth neuron in a hidden layer, the function $\mathrm{g}(\cdot)$ is the activation function in the neuron, n is the number of inputs coming from the previous hidden layer to the current neuron, w_{ij} is the weight of the connection between the ith neuron in the previous layer to the jth neuron in the current layer, b_j is the bias for the jth neuron, and x_i is the output of the ith neuron in the previous hidden layer [40].

In this work, a conventional FCNN model is not used, rather, a multi-headed FCNN input model is employed. This model will use separate fully connected layers to handle each type of input. These separate layers are known as heads. Then, the output of each fully connected layer will be concatenated and processed with more fully connected layers to make the overall output. In this paper, three separate fully connected layers are used to handle each of the different inputs, *i.e.,* time, temperature, and stress. Additionally, some heads also use a dropout layer to reduce overfitting. Fig. 5(a) shows a visual representation of the FCNN model that was used to make the prediction. Fig. 5(a) also presents the number of neurons in each hidden layer and dropout rate in the dropout layers.

A.1.2. Convolutional neural network

CNNs are a type of neural network that use special hidden layers to automatically extract important features from the input data. For instance, in the case of images, these special layers can identify various simple shapes like edges or curves. Larger shapes, like people or objects, can be recognized by continuously applying these layers to the data [41]. Because of these special layers, CNN has had a lot of success in the area of image classification. In addition to image classification, CNN can be adapted to time series forecasting since its hidden layers are also suitable for time series data [38]. In CNN, the special hidden layers are called convolution and pooling layers. Overall, the general CNN model is composed of an input layer, one or more convolution and pooling layers, a fully connected layer, and an output layer [42]. Fig. 11 shows a visual representation of a general CNN. In Fig. 11, the first box represents a large matrix. This input matrix goes through convolution and then pooling. This process can be repeated several times. After going through convolution and pooling, the resulting matrices get flattened and put through a fully connected layer to calculate the output.

The convolution, pooling, and flattening processes used in CNN are described in Eqs. (7), (9), and (11). The convolution process in CNN can vary depending on the input data. For image data, usually two-dimensional convolution is used, but for time series data, one-dimensional convolution is applied. The dimensions in the convolution process refer to the dimensions of the matrix the input data will be convolved with. Eq. (7) shows how the CNN handles one dimensional convolutions. In Eq. (7), the input matrix $\mathbf{A} \in \mathcal{R}^{n\times m}$ is convolved with matrix $\mathbf{W} \in \mathcal{R}^{p\times 1}$ to produce matrix $\mathbf{B} \in \mathcal{R}^{q\times m}$,

$$\begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \cdots & a_{1,m} \\ a_{2,1} & a_{2,2} & a_{2,3} & \cdots & a_{2,m} \\ a_{3,1} & a_{3,2} & a_{3,3} & \cdots & a_{3,m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & a_{n,3} & \cdots & a_{n,m} \end{pmatrix} * \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_p \end{pmatrix} = \begin{pmatrix} b_{1,1} & b_{1,2} & b_{1,3} & \cdots & b_{1,m} \\ b_{2,1} & b_{2,2} & b_{2,3} & \cdots & b_{2,m} \\ b_{3,1} & b_{3,2} & b_{3,3} & \cdots & b_{3,m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ b_{q,1} & b_{q,2} & b_{q,3} & \cdots & b_{q,m} \end{pmatrix} \tag{7}$$

where the symbol $*$ is the convolution operation, m is the number of features in $\mathbf{A}$, n is the original length of $\mathbf{A}$, and q is the length of the data after convolution. The parameter p is often referred as the kernel size and the numbers inside $\mathbf{W}$ are determined through the CNN learning process. The result of the convolution process is matrix $\mathbf{B} \in \mathcal{R}^{q\times m}$, which is referred to as feature map. The values of $\mathbf{B}$ are calculated through Eq. (8),

$$b_{i,j} = \sum_{r=1}^{p} a_{r+i-1,j} \cdot w_r. \tag{8}$$

In CNN, the convolution process does not generate only one feature map, but rather it creates multiple feature maps made with different values in $\mathbf{W}$. This is why there are multiple matrices after the convolution operation in Fig. 11.

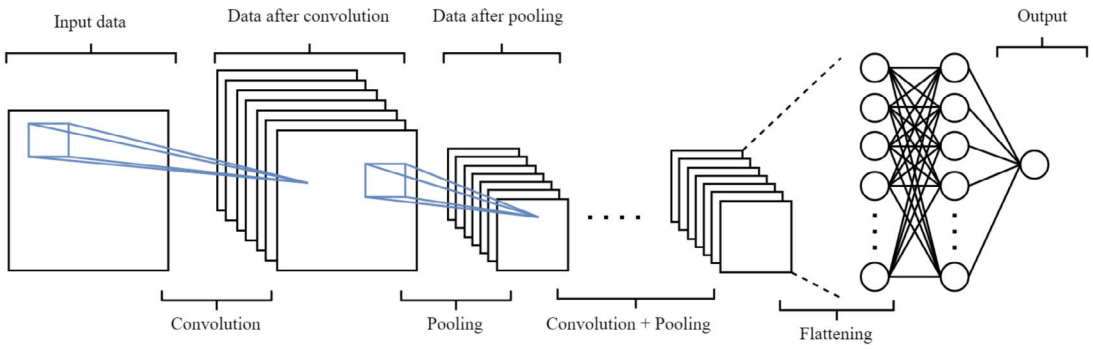

Fig. 11. General CNN.

After convolution, the data is put through a pooling layer. The pooling process can reduce the input matrix even further by selecting certain values inside matrix **B**. Usually, a process called max pooling is used in the pooling layer. Eqs. (9) and (10) show the pooling process when using max pooling. Eq. (9) shows that the matrix made after convolution, **B**, gets transformed into a new matrix $\mathbf{C} \in \mathcal{R}^{l \times m}$,

$$\begin{pmatrix} b_{1,1} & b_{1,2} & b_{1,3} & \cdots & b_{1,m} \\ b_{2,1} & b_{2,2} & b_{2,3} & \cdots & b_{2,m} \\ b_{3,1} & b_{3,2} & b_{3,3} & \cdots & b_{3,m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ b_{q,1} & b_{q,2} & b_{q,3} & \cdots & b_{q,m} \end{pmatrix} \xrightarrow{Pooling} \begin{pmatrix} c_{1,1} & c_{1,2} & c_{1,3} & \cdots & c_{1,m} \\ c_{2,1} & c_{2,2} & c_{2,3} & \cdots & c_{2,m} \\ c_{3,1} & c_{3,2} & c_{3,3} & \cdots & c_{3,m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_{l,1} & c_{l,2} & c_{l,3} & \cdots & c_{l,m} \end{pmatrix} \tag{9}$$

where l is the length of the data after pooling, and all the elements in **C** are calculated through Eq. (10),

$$c_{i,j} = \max\{b_{(i-1)k+1,j}, \ldots, b_{i \cdot k,j}\}. \tag{10}$$

Eq. (10) finds the maximum value within a group of elements inside **B** by using the $\max(\cdot)$ function. Specifically, it finds the maximum number between $b_{(i-1)k+1,j}$ and $b_{i \cdot k,j}$ where the first index of $b_{(i-1)k+1,j}$ increments by 1 until it gets to $b_{i \cdot k,j}$ and k is a known parameter called pooling size.

After the pooling process, the data can be put through more convolution and pooling layers to continuously extract features. Once that is finished, the data gets flattened as shown in Eq. (11),

$$\begin{pmatrix} c_{1,1} & c_{1,2} & c_{1,3} & \cdots & c_{1,m} \\ c_{2,1} & c_{2,2} & c_{2,3} & \cdots & c_{2,m} \\ c_{3,1} & c_{3,2} & c_{3,3} & \cdots & c_{3,m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_{l,1} & c_{l,2} & c_{l,3} & \cdots & c_{l,m} \end{pmatrix} \xrightarrow{Flattening} \begin{pmatrix} c_{1,1} \\ c_{1,2} \\ \vdots \\ c_{1,m} \\ c_{2,1} \\ c_{2,2} \\ \vdots \\ c_{2,m} \\ \vdots \\ c_{l,1} \\ c_{l,2} \\ \vdots \\ c_{l,m} \end{pmatrix}. \tag{11}$$

After flattening, the data goes through a fully connected layer to calculate the output. The process of the fully connected layer here are the same as described in Appendix A.1.1.

The CNN model used in this paper begins with an input layer that takes one matrix including sequences of time, temperature, and stress values. Following the input layer are four pairs of convolution and pooling layers that find the key characteristics of the input data. Finally, the output of the previous layers is flattened and passed through a fully connected layer that interprets the data and calculates the output. Fig. 5(b) shows a visual representation of the CNN model used in this work. Fig. 5(b) also presents the kernel size and number of filters in the convolution layers, the pool size in the pooling layers, and number of neurons in the fully connected layer.

A.1.3. Long short-term memory

LSTM is an advanced recurrent neural network (RNN) that uses special layers to understand long-term temporal dependencies. The special layers, called LSTM layers, contain various LSTM units capable of storing a large sequence of information that is used to make a prediction. Because of the LSTM units, LSTM overcomes the problems present in RNN like gradient explosion, gradient disappearance, and insufficient long-term memory [43,44]. LSTM's ability to memorize allows it to understand the temporal dependencies present in time series data. This characteristic makes LSTM useful for problems where sequences are present, like natural language processing [45] and time series forecasting [46]. Fig. 12(a) shows a visual representation of the general LSTM and Fig. 12(b) shows a more detailed representation of how data flows through an LSTM unit in this neural network.

The main process that occurs in LSTM revolves around controlling the amount of information retained from one LSTM unit to the next. This information is stored in the cell state, and to control the data in it, LSTM uses a series of gates called forget gate, input gate, and output gate. The forget gate removes data from the cell state, the input gate adds new data to the cell state, and the output gate uses the cell state to decide what the LSTM unit is going to output. In Fig. 12(b), the inputs and outputs of an LSTM unit are shown, where x_j is the data coming from input j, h_j and h_{j-1} are the output of the current and previous LSTM unit, respectively, and C_j and C_{j-1} are the current and previous cell states, respectively. The equations in Eq. (12) determine how each gate is affected and how the LSTM units calculates h_j and C_j,

$$\begin{aligned} f_j &= \sigma(W_f \cdot [h_{j-1}, x_j] + b_f) \\ i_j &= \sigma(W_i \cdot [h_{j-1}, x_j] + b_i) \\ C_j &= f_j \times C_{j-1} + (i_j) \times \tanh(W_f \cdot [h_{j-1}, x_j] + b_c) \\ o_j &= \sigma(W_o \cdot [h_{j-1}, x_j] + b_o) \\ h_j &= o_j \times \tanh(C_j) \end{aligned} \tag{12}$$

where f_j, i_j, C_j, and o_j are the output of the forget gate, input gate, cell state, and output gate; W_f, W_i, and W_o are the weights of the forget gate, input gate, and output gate; b_f, b_i, b_c, and b_o are the biases of the forget gate, input gate, cell state and output gate; $[h_{j-1}, x_j]$ is the concatenation between the data of the previous output and the current input; the symbol $\cdot$ is element wise product; the symbol $\times$ is regular multiplication; $\sigma(\cdot)$ and $\tanh(\cdot)$ correspond to the sigmoid and hyperbolic tangent function [47].

In this work, a simple LSTM model is used. It contains an input layer, a dropout layer, an LSTM layers, fully connected layers, and an output layer. The LSTM model takes a matrix containing a series of time, temperature, and stress values as an input and passes it through the LSTM layer. The LSTM layer learn the temporal dependencies within the data, and the output layer converts this information into strain values. Fig. 5(c) shows a visual representation of the LSTM model that presents the number of LSTM units in each LSTM layer and dropout rate in the dropout layer.

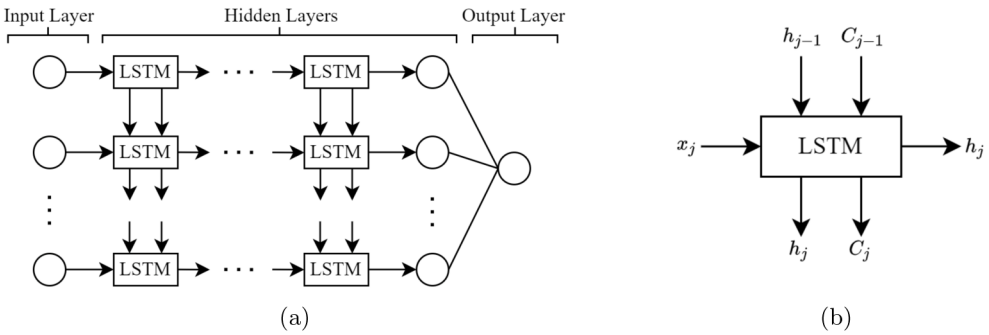

Fig. 12. (a) General LSTM process. (b) Detailed representation of how data is transferred in a single LSTM cell.

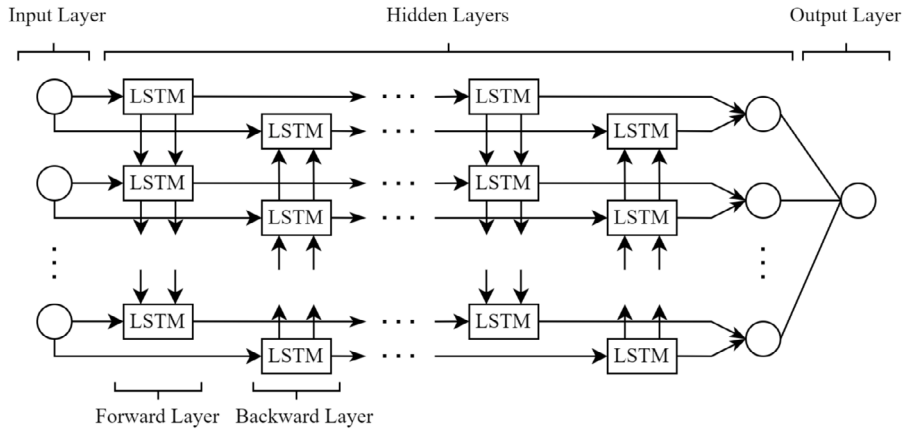

Fig. 13. General BiLSTM.

A.1.4. Bidirectional LSTM

BiLSTM is an advanced version of LSTM capable of using both past and future information to make a prediction. The previously mentioned LSTM model was only capable of using past data to create an output and ignored future data in the prediction process. By using both past and future data, BiLSTM views a larger temporal spectrum and uses it to make the prediction [48]. BiLSTM can use both past and future data by viewing the inputs both forward and backwards, which means that the model will view the inputs from beginning to end and from end to beginning [48,49]. A visual representation of a general BiLSTM is shown in Fig. 13. BiLSTM uses the same LSTM units as the conventional LSTM, where the internal operations of these units can be found through Eq. (12).

The BiLSTM model that was implemented is similar to that of the LSTM model. The difference between the two is the hidden layers change from LSTM layers to BiLSTM layers. Fig. 6(a) shows a visual representation of the BiLSTM model that presents the number of BiLSTM units in each BiLSTM layer.

A.1.5. CNN-LSTM

The CNN-LSTM neural network is a neural network model that uses a combination of the special layers in CNN and LSTM [50]. With CNN-LSTM, the important features can be extracted from the input data using CNN's convolution and pooling layers, and the temporal dependencies can be found through LSTM's layers [51]. Fig. 14 shows a visual representation of a general CNN-LSTM, where the equations discussed in the CNN and LSTM sections can be applied to calculate how the model works.

The CNN-LSTM model that we used, inputted a series of time, temperature, and stress values to convolution layers followed by a pooling layers to extract important information from the data. Following the CNN portion, an LSTM layer was applied, then the data from the LSTM layer was processed to calculate the output. Fig. 6(b) shows a visual representation of the CNN-LSTM model used in this work. Fig. 6(b) presents the kernel size and number of filters in the convolution layers, the pool size in the pooling layer, and the number of LSTM units in each LSTM layer.

A.1.6. CNN bidirectional LSTM

The CNN-BiLSTM model uses a similar concept to CNN-LSTM. Compared to CNN-LSTM, CNN-BiLSTM takes advantage of the more advanced LSTM, BiLSTM [52]. Thus, this model uses convolution and pooling layers to extract important features from the data, and it uses a BiLSTM layer that views a large time spectrum to understand the temporal dependencies of the data and make the prediction. Fig. 15 shows a visual representation of CNN-BiLSTM, where the equations shown in the CNN and LSTM sections can be applied to calculate how the data is transformed thought the neural network.

The workflow in the CNN-BiLSTM model is similar to that of CNN-LSTM. CNN-BiLSTM passes a series of time, temperature, and stress values through convolution and pooling layers. Afterwards, a BiLSTM layer is applied. Following the BiLSTM layer, the data is processed to generate an output. Fig. 6(c) shows a visual representation of the CNN-BiLSTM model used in this work. Fig. 6(c) presents the kernel size and number of filters in the convolution layers, the pool size in the pooling layer, and the number of BiLSTM units in each BiLSTM layer.

A.1.7. Convolutional LSTM

ConvLSTM is another advanced version of LSTM that is designed to deal with spatiotemporal data [53]. The ConvLSTM model revolves around the ConvLSTM layer. This layer has a similar purpose as the

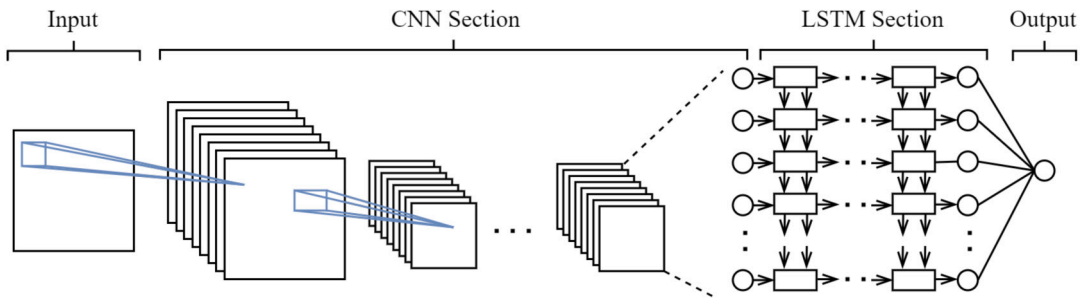

Fig. 14. General CNN-LSTM.

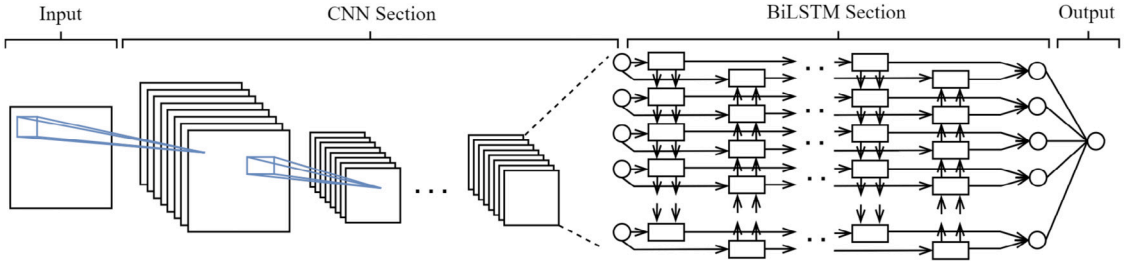

Fig. 15. General CNN-BiLSTM.

LSTM layer but replaces its internal operations with convolution operations [47]. Through the use of convolutions within LSTM, the ConvLSTM layer becomes useful to deal with spatiotemporal data like a sequence of frames in a video. In addition to spatiotemporal data, ConvLSTM can also be used for time series forecasting problems [54]. Eq. (13) shows the equations for LSTM but with the convolution operation ($*$) instead of pointwise multiplication ($\cdot$),

$$\begin{aligned} f_j &= \sigma(W_f * [h_{j-1}, x_j] + b_f) \\ i_j &= \sigma(W_i * [h_{j-1}, x_j] + b_i) \\ C_j &= f_j \times C_{j-1} + i_j \times \tanh(W_f * [h_{j-1}, x_j] + b_c) \\ o_j &= \sigma(W_o * [h_{j-1}, x_j] + b_o) \\ h_j &= o_j \times \tanh(C_j). \end{aligned} \tag{13}$$

The ConvLSTM model starts with an input layer that takes in a matrix containing a series of time, temperature, and stress values. Then, it goes through a ConvLSTM layer where the data is processed and passed to a flatten layer. After the data is flattened, the prediction is calculated by a final output layer. Fig. 6(d) shows a visual representation of the ConvLSTM model used in this work. Fig. 6(d) presents the kernel size and number of filters for the convolution operations in the ConvLSTM layer.

A.1.8. Ensemble

An ensemble is a technique that combines multiple deep learning models to make a prediction. Ensemble neural networks have the ability to reduce the variance of singular neural networks and are often capable of making better predictions than the singular neural networks inside the ensemble [55]. To make the ensemble neural network, the three models presented in the core models section are combined and used to predict the resulting strain based on time, temperature, and stress values. Each model within the ensemble will be trained individually with the same training data. After training, each model will be used to make a prediction. Finally, the predictions from each model will be averaged to produce an output.

A.2. Diebold–Mariano test

When comparing forecasts, it is common to point out a better performing model by comparing the errors made by each prediction. In this case, it was found that the prediction error of CNN's forecast was lower than the rest of the forecasts made with the other neural networks. However, there exists the possibility that CNN's prediction error was lower than the other models' predictions because of chance and not because CNN's forecast is more accurate. To evaluate the previous statement, CNN's prediction would be compared with the prediction made by each DL model. This comparison determines if there is a significant difference between CNN's prediction and another prediction from one of the other DL models used. To make this comparison the Diebold–Mariano (DM) Test was used [56]. The DM test compares two forecasts to determine if the difference in error is significant or due to chance [57].

In the following, we will show that the reason why CNN's forecast error was lower than that of the rest of the models is not because of chance. We begin with the hypothesis assumed by the DM test, that two forecasts (CNN's forecasts and a forecast from another DL model) have equal error [57]. To prove the initial statement, the hypothesis needs to be rejected. We start by assuming the hypothesis is true and calculate the probability of obtaining the observed results, *i.e,* CNN's prediction error is lower than the other prediction errors. If the probability is high (it is likely to obtain the observed results assuming both predictions have equal error), then the hypothesis fails to be rejected. If the probability is low (it is unlikely to obtain the observed results assuming both predictions have equal error), the hypothesis is rejected. In the case where the hypothesis fails to be rejected, then there is no significant difference between the CNN's prediction and the prediction made by other models, which means that CNN's prediction is not necessarily better than the prediction it was compared with. Otherwise, CNN's prediction is indeed better than the prediction it was compared with.

To use the DM test, the DM statistic is calculated and used to determine a p-value. The p-value shows the likelihood of obtaining the observed results in the case that the hypothesis is true. In addition to the p-value, a threshold is needed to define what a high or low probability is. This threshold is called significance level denoted as α. In this work, α is 5%, which is common in practice. The p-value is calculated using Eq. (14),

$$p\text{-value} = 2\,\mathrm{tcdf}(-|DM|) \tag{14}$$

where $\mathrm{tcdf}(\cdot)$ is the Student's t-distribution function and DM is the DM statistic. To calculate the DM statistic, Eqs. (15), (16), and (17) are used,

$$\mathrm{DM} = \frac{\bar{d}}{\sqrt{\frac{1}{N}\left(\gamma_0 + 2\sum_{k=1}^{h-1}\gamma_k\right)}} \tag{15}$$

$$\bar{d} = \frac{1}{N}\sum_{i=1}^{N} d_i \tag{16}$$

$$\gamma_k = \frac{1}{N}\sum_{i=1}^{N}(d_i - \bar{d})(d_{i-k} - \bar{d}) \tag{17}$$

where N is the number of data points; h is the forecast horizon; $\bar{d}$ is the mean of d; γ_k is the autocorrelation function; and γ_0 is the variance of d. The difference between the squared errors, d, is calculated as $d = (e_1)^2 - (e_2)^2$ where e_1 is the difference between the ground truth and the first forecast $e_1 = y - \hat{y_1}$ and e_2 is the difference between the ground truth and the second forecast $e_2 = y - \hat{y_2}$. In this work, we used h as 1, which is common when using the DM test. If h is 1, then there is no need to calculate γ_k.

To perform this test, we compared the prediction of CNN with the other predictions. After doing so, we found that all the calculated p-values were lesser than 5%. At this point, we reject the initial hypothesis, which indicates that the prediction made by the CNN model was indeed better and not due to chance.

References

[1] W.M. Huang, et al., Shape memory materials, Mater. Today 13 (2010) 54–61.
[2] H. Meng, G. Li, A review of stimuli-responsive shape memory polymer composites, Polymer 54 (2013) 2199–2221.
[3] W. Sokolowski, et al., Medical applications of shape memory polymers, Biomed. Mater. 2 (2007) S23–S27.
[4] Y. Liu, et al., Shape memory polymers and their composites in aerospace applications: A review, Smart Mater. Struct. 23 (2014) 023001.
[5] S. Thakur, Shape memory polymers for smart textile applications, in: Bipin Kumar, Suman Thakur (Eds.), Textiles for Advanced Applications, IntechOpen, Rijeka, 2017.
[6] C. Yan, et al., Machine learning assisted discovery of new thermoset shape memory polymers based on a small training dataset, Polymer 214 (2021) 123351.
[7] K. Guo, et al., Artificial intelligence and machine learning in design of mechanical materials, Mater. Horiz. 8 (2021) 1153–1172.
[8] J. Wei, et al., Machine learning in materials science, InfoMat 1 (2019) 338–358.
[9] JW. Lee, et al., Dirty engineering data-driven inverse prediction machine learning model, Sci. Rep. 10 (2020).
[10] C. Yan, G. Li, Machine learning framework for polymer discovery, in: Reference Module in Materials Science and Materials Engineering, 2021.
[11] S. Wu, et al., Machine-learning-assisted discovery of polymers with high thermal conductivity using a molecular design algorithm, Npj Comput. Mater. 5 (2019).
[12] L. Miccio, G. Schwartz, From chemical structure to quantitative polymer properties prediction through convolutional neural networks, Polymer 193 (2020) 122341.
[13] L. Tao, G. Chen, Y. Li, Machine learning discovery of high-temperature polymers, Patterns 4 (2021) 100225.
[14] C. Yan, X. Feng, G. Li, From drug molecules to thermoset shape memory polymers: A machine learning approach, ACS Appl. Mater. Interfaces 13 (2021) 60508–60521.
[15] A. Mannodi-Kanakkithodi, et al., Machine learning strategy for accelerated design of polymer dielectrics, Sci. Rep. 6 (2016).
[16] C. Yan, Q. Yang, G. Li, A phenomenological constitutive model for semicrystalline two-way shape memory polymers, Int. J. Mech. Sci. 177 (2020) 105552.
[17] G. Li, W. Xu, Thermomechanical behavior of thermoset shape memory polymer programmed by cold-compression: Testing and constitutive modeling, J. Mech. Phys. Solids 59 (6) (2011) 1231–1250.
[18] A. Shojaei, G. Li, Thermomechanical constitutive modelling of shape memory polymer including continuum functional and mechanical damage effects, Proc. Royal Soc. A 470 (2014) 20140199.
[19] Y. Liu, et al., Thermomechanics of shape memory polymers: UNIAXIAL experiments and constitutive modeling, Int. J. Plast. 22 (2006) 279–313.
[20] H.J. Qi, et al., Finite deformation thermo-mechanical behavior of thermally induced shape memory polymers, J. Mech. Phys. Solids 256 (2008) 1730–1751.
[21] Q. Yang, G. Li, Temperature and rate dependent thermomechanical modeling of shape memory polymers with physics based phase evolution law, Int. J. Plast. 80 (2016) 168–186.
[22] L. Dai, C. Tian, R. Xiao, Modeling the thermo-mechanical behavior and constrained recovery performance of cold-programmed amorphous shape-memory polymers, Int. J. Plast. 127 (2020) 102654.
[23] H. Tobushi, et al., Thermomechanical constitutive model of shape memory polymer, Mech. Mater. 33 (2001) 545–554.
[24] M. Heuchel, et al., Relaxation based modeling of tunable shape recovery kinetics observed under isothermal conditions for amorphous shape-memory polymers, Polymer 51 (2010) 6212–6218.
[25] L. Haibao, W. Wei Min, On the origin of the Vogel–Fulcher–Tammann law in the thermo-responsive shape memory effect of amorphous polymers, Smart Mater. Struct. 22 (2013).
[26] F. Masi, et al., Thermodynamics-based artificial neural networks for constitutive modeling, J. Mech. Phys. Solids 147 (2021) 104277.
[27] K. Wang, W.C. Sun, A multiscale multi-permeability poroplasticity model linked by recursive homogenizations and deep learning, Comput. Methods Appl. Mech. Engrg. 334 (2018) 337–380.
[28] A.M. Tartakovsky, et al., Physics-informed deep neural networks for learning parameters and constitutive relationships in subsurface flow problems, Water Resour. Res. 56 (2020).
[29] T. Furukawa, G. Yagawa, Implicit constitutive modelling for viscoplasticity using neural networks, Internat. J. Numer. Methods Engrg. 43 (1998) 195–219.
[30] T. Kirchdoerfer, M. Ortiz, Data-driven computational mechanics, Comput. Methods Appl. Mech. Engrg. 304 (2016) 81–101.
[31] R. Ibáñez, et al., A manifold learning approach to data-driven computational elasticity and inelasticity, Arch. Comput. Methods Eng. 25 (2017) 47–57.
[32] L. Lu, J. Cao, G. Li, Giant reversible elongation upon cooling and contraction upon heating for a crosslinked cis poly(1,4-butadiene) system at temperatures below zero celsius, Sci. Rep. 8 (2018).
[33] K. Westbrook, et al., Constitutive modeling of shape memory effects in semicrystalline polymers with stretch induced crystallization, J. Eng. Mater. Technol. 132 (2010).
[34] R.B. Hall, I.J. Rao, H.J. Qi, Thermodynamics and thermal decomposition for shape memory effects with crystallization based on dissipation and logarithmic strain, Mech. Time-Dependent Mater. 18 (2014) 437–452.
[35] O. Dolynchuk, I.S. Kolesov, H.-J. Radusch, Thermodynamic description and modeling of two-way shape-memory effect in crosslinked semicrystalline polymers†, Polym. Adv. Technol. 25 (2014) 1307–1314.
[36] G. Scalet, et al., A one-dimensional phenomenological model for the two-way shape-memory effect in semi-crystalline networks, Polymer 158 (2018) 130–148.
[37] K. Sree, C. Bindu, Data analytics: Why data normalization, Int. J. Eng. Technol. (UAE) 7 (2018) 209–213.
[38] P. Lara-Benítez, M. Carranza-García, J.C. Riquelme, An experimental review on deep learning architectures for time series forecasting, Int. J. Neural Syst. 31 (2021) 2130001.
[39] F. Murtagh, Multilayer perceptrons for classification and regression, Neurocomputing 2 (1991) 183–197.
[40] C. Bishop, Neural Networks for Pattern Recognition, 1995.
[41] M. Hussain, J.J. Bird, D.R. Faria, A study on CNN transfer learning for image classification, in: Advances in Intelligent Systems and Computing, 1918, pp. 191–202.
[42] Y. Lecun, Y. Bengio, Convolutional networks for images, speech, and time-series, in: M.A. Arbib (Ed.), The Handbook of Brain Theory and Neural Networks, MIT Press, 1995.
[43] F.A. Gers, Learning to forget: Continual prediction with LSTM, in: 9th International Conference on Artificial Neural Networks, ICANN '99, 1999.
[44] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (1997) 1735–1780.
[45] L. Yao, Y. Guan, An improved LSTM structure for natural language processing, in: 2018 IEEE International Conference of Safety Produce Informatization, IICSPI, 2018, pp. 565–569.
[46] S. Siami-Namini, N. Tavakoli, A. Siami Namin, A comparison of ARIMA and LSTM in forecasting time series, in: 2018 17th IEEE International Conference on Machine Learning and Applications, ICMLA, 2018, pp. 1394–1401.
[47] H. Qiao, et al., A time-distributed spatiotemporal feature learning method for machine health monitoring with multi-sensor time series, Sensors 18 (2018).
[48] H. Jahangir, et al., Deep learning-based forecasting approach in smart grids with microclustering and bidirectional LSTM network, IEEE Trans. Ind. Electron. 68 (9) (2021) 8298–8309.
[49] T. Thireou, M. Reczko, Bidirectional long short-term memory networks for predicting the subcellular localization of eukaryotic proteins, IEEE/ACM Trans. Comput. Biol. Bioinform. 4 (2007) 441–446.
[50] J. Wang, et al., Dimensional sentiment analysis using a regional CNN-LSTM model, in: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), Association for Computational Linguistics, Berlin, Germany, 2016, pp. 225–230.
[51] W. Lu, et al., A CNN-LSTM-based model to forecast stock prices, Complexity 2020 (2020) 1–10.
[52] V. Passricha, R. Aggarwal, A hybrid of deep CNN and bidirectional LSTM for automatic speech recognition, J. Intell. Syst. 29 (2020) 1261–1274.

[53] X. Shi, et al., Convolutional LSTM network: A machine learning approach for precipitation nowcasting, in: C. Cortes, et al. (Eds.), Advances in Neural Information Processing Systems, Vol. 28, Curran Associates, Inc., 2015.
[54] D. Wang, Y. Yang, S. Ning, DeepSTCL: A deep spatio-temporal ConvLSTM for travel demand prediction, in: 2018 International Joint Conference on Neural Networks, IJCNN, 2018, pp. 1–8.
[55] S. Zhang, M. Liu, J. Yan, The diversified ensemble neural network, in: H. Larochelle, et al. (Eds.), Advances in Neural Information Processing Systems, Vol. 33, Curran Associates, Inc., 2020, pp. 16001–16011.
[56] F. Diebold, R. Mariano, Comparing predictive accuracy, J. Bus. Econom. Statist. 20 (2002) 134–144.
[57] F. Diebold, Comparing predictive accuracy, twenty years later: A personal perspective on the use and abuse of Diebold–Mariano tests, J. Bus. Econom. Statist. 33 (2015) 1.