# AI for Learning Deformation Behavior of a Material: Predicting Stress-Strain Curves 4000x Faster Than Simulations

Yuwei Mao*, Shahriyar Keshavarz†, Vishu Gupta*, Andrew C.E. Reid†,
Wei-keng Liao*, Alok Choudhary*, Ankit Agrawal*
*ECE department, Northwestern University, Evanston, USA
{yuweimao2019,vishugupta2020}@u.northwestern.edu, {wkliao,choudhar,ankitag}@eecs.northwestern.edu
†National Institute of Standards and Technology, Gaithersburg, USA
{shahriyar.keshavarz, andrew.reid}@nist.gov

***Abstract*—Stress-strain curves are important representations of a given material's mechanical properties, which depend primarily on the orientation of the individual crystals in the microstructure. Generating stress-strain curves from numerical methods such as the crystal plasticity finite element (CPFE) simulations is computationally intensive. As a result, it is difficult to generate complete stress-strain curves for all possible orientations of a material. In this work, we propose a bilinear stress-strain curve prediction framework for metallic alloys by integrating supervised and unsupervised deep learning methods via transfer learning principles. As a specific case-study, we focus on predicting stress-strain curves of Nickel (Ni)-based superalloys that have important applications in aerospace industry. Using a small training set of just 100 complete stress-strain curves (4,000 strain steps each) of different orientations generated by CPFE simulation code, we were able to build a model that could accurately predict stress-strain curves ( $<2\,\%$ error) using simple features that could be obtained by running the CPFE simulation for just a single strain step. The proposed model can thus predict the complete stress-strain curve for a given orientation of Ni-based superalloys in a fraction of a second, which amounts to a speedup of over 4000x as compared to the simulation.**

***Index Terms*—stress-strain, encoder, decoder, neural network.**

## I. Introduction

Deformation is the action or process of distorting. When a force is applied to a material, the material will either compress or stretch as a response to the force. In mechanics, the force applied to a unit area is called stress. The extent of stretching or compressing (as a response to stress) is called strain. Deformations can be elastic or plastic based on what happens after the stress is released. Elastic deformation is the deformation that disappears upon removal of the external forces causing the alteration and the stress associated with it. Plastic deformation is a permanent deformation or change in the shape of a solid body without fracture under the action of a sustained force.

Stress-strain curves are representations of the deformation behavior for materials, consisting of an elastic and a plastic region separated by a yield point for most materials. Polycrystalline materials are composed of many crystallites of varying size and orientation. The deformation morphologies of polycrystalline materials are strongly dependent on the crystallographic orientations, which means stress-strain curves depend on the orientations. Analyzing stress-strain curves could help scientists understand the deformation behavior and learn more information about the relationship between structure and property for polycrystalline materials. Thus, generating stress-strain curves according to different crystallographic orientations for materials is a very important problem in materials science.

The crystal plasticity finite element (CPFE) [1] simulations can be used to generate stress-strain curves. Figure 1 shows some example stress-strain curves of Ni-based superalloys generated by CPFE simulations. The stress is plotted on the y-axis and its corresponding strain is on the x-axis. The curve is very steep at the beginning (the elastic zone), and after the bend point the curve slope decreases (the plastic zone). However, CPFE simulation is computationally expensive and time-consuming, especially if a stress-strain behavior of sufficient length needs to be explored. In this study, we use artificial intelligence methods to rapidly generate this type of stress-strain curves of Ni that consists of two almost straight lines, where the intersection of two lines is the yield point.

With the fast development of artificial intelligence and machine learning (AI/ML) techniques, and the increasing availability of data from the first three paradigms of science (experiments, theory, and simulations), the fourth paradigm of science, i.e., data-driven science and discovery, is playing an important role in materials science [1]. Many machine learning and deep learning techniques have been extensively used in many materials science applications [2]–[8] to enhance materials property prediction, discovery, and design. Such models do not require significant human intervention or knowledge, learn relationships efficiently relative to the input design space,

[1]Certain commercial equipment, instruments, or materials (or suppliers, or software, ...) are identified in this paper to foster understanding. Such identification does not imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the materials or equipment identified are necessarily the best available for the purpose.

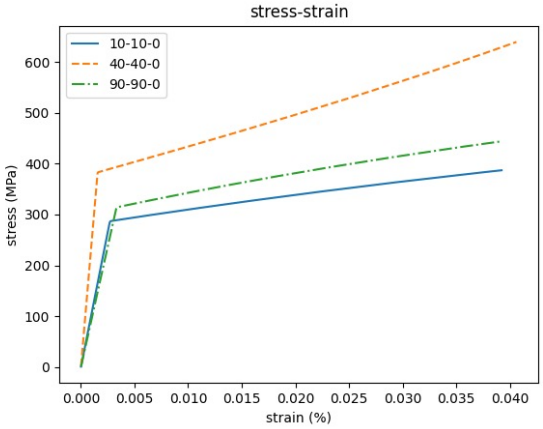



Fig. 1: Example stress-strain curves of Ni-based superalloys for orientations “10-10-0”, “40-40-0”, and “90-90-0”.

and can be generalized to different systems.

Recently, researchers have tried using machine learning models to predict stress-strain curves at a much higher speed. Yang et al. [9] proposed an artificial neural network (ANN) model combined with principal component analysis (PCA) to predict the stress-strain curve of binary composites. Janab et al. [10] used a genetic algorithm and ANN to predict stress-strain values of AA5182-O aluminum alloy sheets. Ali et al. [11] employed ANN to estimate the stress-strain curve of AA6063-T6 aluminum alloy under non-proportional loading conditions. Koenuma et al. [12] used the deep learning method with a rate-dependent crystal plasticity finite element method formulation to predict the stress-strain curve of aluminum alloy sheets. In [13], Yamanaka et al. trained two DNNs to estimate biaxial stress-strain curves of sheet metals from their underlying microstructural features. Merayo et al. [14] accurately predict the yield stress (YS) and the ultimate tensile strength (UTS) of aluminum alloys based on Brinell hardness data using the ANN model.

In [15], Setti et al. used different architectures of neural networks to predict the stress-strain curve of near beta titanium alloy. Setti et al. [15] used the ANN approach to predict the stress-strain curve of titanium alloy. In ANN training, the module volume fraction of $\alpha$ and strain were employed as input and stress as output. In this study, we focus on rapidly predicting bilinear stress-strain curves for metallic alloys with crystalline microstructures (with the specific case study on Ni-based superalloys) using only simple features obtained by running the CPFE simulation for a single strain step. Ni-based superalloys are widely used in the aerospace industry, especially in hot sections of turbine engine components, such as blades, disks, casings and liners due to their excellent mechanical properties [16]. We thus select Ni-based superalloys as the specific case study in this work.

A stress-strain curve prediction framework is proposed in this study. The proposed framework only needs a small number of stress-strain curves with 4,000 strain steps to predict stress-strain curves for other orientations. The framework first fits the stress-strain curve as a bilinear function to get four parameters. A multi-layer feed-forward perceptron (MLP) is trained using these parameters as labels and intermediate data obtained by running the CPFE simulation for a single initial strain step as features. After training, the model can be used to predict these parameters, and generate bilinear stress-strain curves according to the predicted parameters.

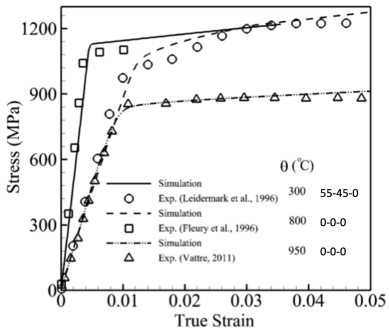
Moreover, since the available labeled training data is too small, we generate a larger unlabeled dataset using easy-to-obtain features from more orientations for unsupervised learning to make use of the orientation-based information. We train an autoencoder model on this unlabeled dataset. The learned weights are then transferred as initial weights to train a new MLP model. The proposed framework is cross-validated on 100 stress-strain curves of different orientations generated by CPFE simulations. The experimental results demonstrate that the proposed framework has significant computational savings while still maintaining the accuracy of the prediction. In particular, we believe that the ability to quickly and accurately predict complete stress-strain curves from simple and easy-to-obtain input features is an attractive aspect of the proposed framework. But since the material use-case, simulation to generate the data, and the type of input features used in this work are different from previous works, they are not directly comparable.

## II. Crystal Plasticity Background

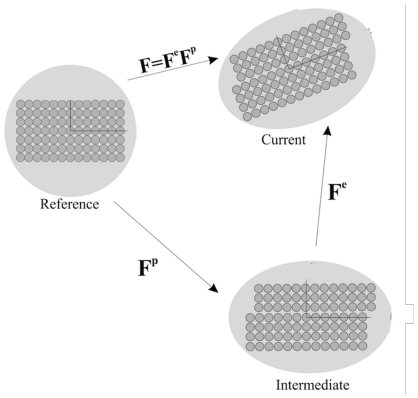
### A. Crystal Plasticity Finite Element Method

A crystal orientation is given by Miller Indices, $\langle hkl \rangle$, as plane normal vectors. Majority of metal and metallic alloys properties are associated with crystal orientations specifying anisotropic characteristics of these materials. In single crystals, orientation dependence can result in significant discrepancies in mechanical properties. Different crystal orientations are related through Euler angles. For convenience, the crystal orientations in this study are represented by Euler angles as ”$O_x$-$O_y$-$O_z$”. The range of $O_x$ and $O_y$ is [0, 90]. $O_z$ is always equal to zero because of the crystal orientation dependence. Figure 2(a) displays different stress-strain curves for a single crystal Ni-based superalloys in both elastic and plastic regimes for dissimilar orientations of ”0-0-0” and ”55-45-0” from the CPFE simulations and compared to the experimental data.

The elastic-plastic deformation of crystalline multiphase aggregates depends on the direction of loading, i.e. crystals are mechanically anisotropic. The directionality or orientation dependence of the mechanical response of crystals under load is due to the anisotropy of the elastic tensor and to the orientation dependence of the activation of certain crystallographic deformation mechanisms. To address the crystalline anisotropy and size dependence aspects in crystalline materials, crystal plasticity (CP), a proper tool reflecting the anisotropic nature of crystalline materials, along with finite element (FE) analysis is utilized. The CPFE method in the large deformation platform, when performed in three dimensions, has the ability to




(a) The stress-strain curves of a single crystal Ni-based superalloys resulted from the CP code and compared to the experimental data [17].




(b) Framework decomposition into reference, intermediate, and current configurations.

Fig. 2: Crystal plasticity backgrounds

assess and simulate the grain interactions, interface abrupt mechanical transitions, mixed deformation mechanisms, complex boundary conditions, and diverse empirical and physics-based constitutive models. The three-dimensional CPFE method can be applied to generate stress-strain curves for any orientation.

### *B. Crystal Plasticity in Large Deformation Finite Element*

Plasticity theories are applicable primarily to solids experiencing inelastic deformations considerably greater than elastic ones. The crystal plasticity approach with distinguished potentials can be considered for either small or large deformation [18]. While small deformation relations simplify the implementation and simulation process, yet, make the model limited to a small strain range. In the current study, large deformation crystal plasticity formulations are implemented.

In continuum mechanics, deformation gradient tensor $\mathbf{F}$ ( $\mathbf{F} = \frac{d\mathbf{x}}{d\mathbf{X}}$) is considered to map current configuration to reference configuration figure 2(b). Here $\mathbf{X}$ is position of a point in the reference (undeformed) configuration and $\mathbf{x}$ is position of the same point in the current (deformed) configuration. Large strain-description is accommodated through a multiplicative decomposition of the total deformation gradient, $\mathbf{F}$, into the elastic deformation gradient, $\mathbf{F}^e$, which is the deformation component due to the reversible response of the lattice under external loads and displacements as well as rigid-body rotations, and the plastic deformation gradient, $\mathbf{F}^p$, as an irreversible deformation that persists when all external forces and displacements are removed. In this sense, the transformation of the reference state by $\mathbf{F}^p$ leads to the creation of the intermediate configuration. The intermediate configuration signifies the presence of dislocations which produce the permanent shape changes. In addition, due to the rate dependence of most metals and metallic alloys, the rate of deformation gradient needs to be involved through the velocity gradient, $\mathbf{l} = \dot{\mathbf{F}}\mathbf{F}^{-1}$ ($\dot{\mathbf{F}}$ is rate of deformation gradient tensor), as well.

The decomposition framework is demonstrated in figure 2(b), where an atomistic structure in the reference configuration is in the undeformed state. The current configuration displays the deformed structure including both elastic and plastic deformations where the plastic part of displacements in the form of stretch type is calculated in the intermediate configuration. With the presence of plastic deformation, multiplicative decomposition is adopted in order to divide the elastic and plastic parts of the deformation. In fact, an intermediate configuration is introduced to solve the indeterministic equation of $\mathbf{F} = \mathbf{F}^e\mathbf{F}^p$ by calculating $\mathbf{F}^p$. For a given deformation, the plastic part $\mathbf{F}^p$ is calculated using constitutive model via the plastic part of the velocity gradient, $\mathbf{l}^p = \dot{\mathbf{F}}^p(\mathbf{F}^p)^{-1}$. Then, the deformation tensor $\mathbf{F}$ and the plastic strain $\mathbf{F}^p$ could be used to calculate stress and strain values further.

Thus, given the orientation number, $\mathbf{F}$ and $\mathbf{F}^p$ (both $\mathbf{F}$ and $\mathbf{F}^p$ are $3 \times 3$ matrices) are obtained as intermediate data and stress-strain values are obtained as final data at each strain step by CPFE simulations. Then the stress-strain curve can be plotted over the complete strain steps. The more strain steps we want to generate, the longer it takes. In order to understand the complete deformation behavior of the material, 4,000 strain steps are recommended by domain scientists in this study, which can take up to an hour to calculate the stress-strain curve for a single orientation. It is infeasible to generate a large number of stress-strain curves using CPFE simulations directly.

## III. Methods

### *A. Proposed Framework*

Figure 3 shows an overview of the proposed stress-strain prediction framework by combining supervised and unsupervised learning. Two datasets are generated using CPFE simulations: a labeled dataset and an unlabeled dataset. The labeled data are complete stress-strain curves and the corresponding $\mathbf{F}^p$, $\mathbf{F}$ matrices with 4,000 strain steps. Only 100 labeled data

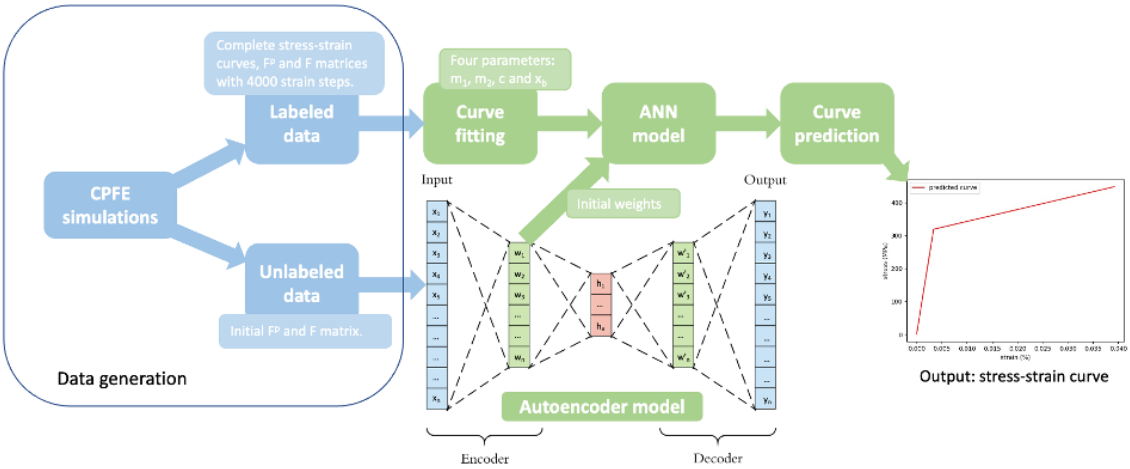



Fig. 3: Stress-strain curve prediction framework based on deep learning methods.

from 100 different orientations could be obtained due to time constraints. We use bilinear functions to describe these 100 stress-strain curves. The bilinear function is defined by four parameters: $m_1, m_2, c$, and $x_b$. These parameters are used as labels to train a MLP model, where the input features are just the initial $\mathbf{F}^p$ and $\mathbf{F}$ matrices. After that, the stress-strain curve can be predicted according to four parameters predicted by the MLP model. In order to take advantage of more information from more orientations without taking too much time, we generate the initial $\mathbf{F}^p$ and $\mathbf{F}$ matrices for more orientations to construct the unlabeled dataset. We take these data as inputs to train an autoencoder model to reconstruct the inputs. The encoder model in the autoencoder model has the same architecture as the MLP model. Then, the weights of the encoder model are transferred and utilized as the initial weights for training the MLP model via transfer learning principles.

### B. Data Generation

In this study, 4,000 steps for each orientation are generated to make sure the stress-strain curves include the deformation behavior of sufficient length. We call the stress-strain curve with 4,000 steps as the complete stress-strain curve. Only 100 complete stress-strain curves are generated as the labeled data due to time constraints. In order to get well-distributed data, we traversed every element from the list {0,10,20,30,40,50,60,70,80,90} for $O_x$ and $O_y$ as the inputs to run CPFE simulations, i.e., a total of 100 orientations. The stress-strain curve for each orientation has 4,000 points or strain steps. Thus, 4,000 $\mathbf{F}$ and $\mathbf{F}^p$ matrices $\{\mathbf{F}_0, \mathbf{F}_1, ....., \mathbf{F}_{3999}\}$ and $\{\mathbf{F}_0^p, \mathbf{F}_1^p, ....., \mathbf{F}_{3999}^p\}$ for each orientation were obtained. $\mathbf{F}_0$ and $\mathbf{F}_0^p$ equals a $3 \times 3$ identity matrix for all orientations, the rest $\mathbf{F}$ and $\mathbf{F}^p$ matrices are different for different orientations. Thus, we take $\mathbf{F}_1$ and $\mathbf{F}_1^p$ as the initial $\mathbf{F}$ and $\mathbf{F}^p$ matrices for each orientation to be used as inputs for the MLP model. Figure 1 shows three example stress-strain curves, where the orientations are "10-10-0", "40-40-0" and "90-90-0".

For generating unlabeled data, we set the number of steps as one to generate the initial $\mathbf{F}$ and $\mathbf{F}^p$ matrices. Because the generation time is much less for only one step, we could generate more data to construct the unlabeled dataset. Here we traverse $O_x$ and $O_y$ from all integers in [0, 90] to obtain 8281 data points from different orientations.

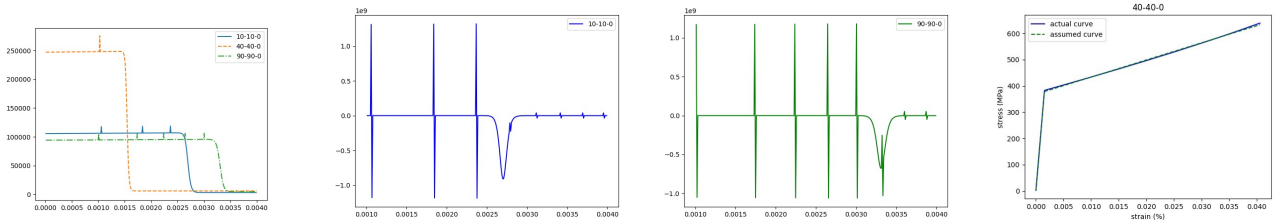
### C. Curve Fitting

In Figure 2(a), it is easy to distinguish the elastic regime (linear and very steep) and the plastic regime, where the curve slope decreases and becomes flatter. There is an obvious rapid change near the yield point. Other stress-strain curves are similar to these three examples according to our observation. Thus, the relationship of stress and strain can be reasonably represented by a bilinear fit over the entire range of 4,000 strain steps. The bilinear approximation of the stress-strain curve consists of two lines that represent, respectively, the elastic behavior and the plastic behavior. These two lines intersect at the yield point. We use the bilinear function as given by equation 1 to represent the stress-strain curve.

$$y = \begin{cases} m_1 x, & x < x_b \\ m_2 x + c, & x \geq x_b \end{cases} \tag{1}$$

It is defined by four parameters: $m_1, m_2, c$, and $x_b$. Note that there are only three independent parameters actually, as knowing any three of these is sufficient to solve for the fourth. If we can get the values of these parameters for each orientation, the corresponding stress-strain curve for this orientation can be plotted. The method for calculating $m_1, m_2, c$, and $x_b$ for these 100 stress-strain curves is introduced in this section.

For a bilinear curve, the slope is $m_1$ before the bend point and $m_2$ after the bend point. The bend point of the bilinear curve is the yield point of the stress-strain curve. However, these stress-strain curves are not really bilinear. Figure 4(a) shows the slope values for every point. In the figure, the slope values are not always the same before the yield point and after the yield point. And there is a segment rather than a point




(a) (b) (c) (d)

Fig. 4: (a) first derivative curves of three stress-strain curves; second derivative curve for stress-strain curve of (b) 10-10-0; (c) 90-90-0; (d) actual stress-strain curve and assumed bilinear stress-strain curve for orientation 40-40-0.

that characterizes the huge difference in slope value. We call this segment the yield segment. In order to have a bilinear fit, we need to set an assumed yield point to represent the yield segment. First, we calculate the second derivative value for every point. Figure 4 (b)(c) show some example second derivative curves. In the second derivative curve, the segments whose values are negative indicate that the slope value is reduced. The longest segments where the second derivative is negative in figure 4 (b)(c) corresponds to segments where the slopes are greatly reduced for orientations "10-10-0" and "90-90-0", respectively, in figure 4 (a). It is easy to see that the longest segments correspond to the yield segments. In figure 4 (a)(b)(c), we could find that there are several peaks besides segments that the slope values are greatly reduced (i.e., the yield segments). Obviously, these peaks are noise from the numerical simulation. The difference between noise peaks and the yield segments is the noise peaks are very short segments. Therefore, the longest segment where the second derivative is negative is the yield segment, and others are noise points. After obtaining the yield segment, we set the middle point of the yield segment as the assumed yield point $(x_a, y_a)$.

After getting the yield point $(x_a, y_a)$, we use the fit function in matrix laboratory (MATLAB) to fit two linear curves $y = m_1 x$ and $y = m_2 x + c$. Then, the intersection point can be calculated by equation 2, which may or may not be identical to the assumed yield point $(x_a, y_a)$.

$$x_b = \frac{c}{m_1 - m_2} \tag{2}$$

We select the intersection point as the yield point $(x_b, y_b)$ for function 1. All four parameters $m_1, m_2, c$, and $x_b$ are obtained. Figure 4(d) shows a comparison between the actual stress-strain curve of orientation "40-40-0" and its bilinear approximation, called assumed curve in this paper. As can be seen, the fit of the bilinear model is good in both lines and the yield point.

### *D. An ANN Model for Curve Prediction*

Artificial neural networks have two key characteristics, the networks have adaptive learning ability and the learned knowledge is stored in the connection weights between neurons. Neural networks have powerful modeling capabilities for regression and classification problems. There are many popular types of neural networks, such as convolutional neural networks (CNN), recurrent neural network (RNN) and generative adversarial network (GAN) [19]. In this paper, a MLP neural network is used to model the parameters of bilinear stress-strain curves. In a feed-forward perceptron network, nodes between adjacent layers are fully connected. The architecture of the network in this paper has one hidden layer with 64 neurons.

The data for each orientation is considered as a data point. It is clear that both $\mathbf{F}$, $\mathbf{F}^p$ and the orientation number can contain the information of stress-strain curves for different orientations. We try several combinations for these features in the experiments to explore the encoding information of these features. Using fewer steps' $\mathbf{F}$ and $\mathbf{F}^p$ matrices is desirable as it can reduce the prediction time. We know that $\mathbf{F}_0$ and $\mathbf{F}_0^p$ are the same for all orientations, so do not contain any information for different stress-strain curves of different orientations. Therefore, we use $\mathbf{F}_1$ and $\mathbf{F}_1^p$ matrices and orientation numbers $O_x$ and $O_y$ as the inputs. All the input features are normalized in the range of [-1, 1]. The parameters $m_1, m_2, c$, and $x_b$ are labels for MLP model. The trained model can be used to predict the parameters $m_1, m_2, c$ and $x_b$. After obtaining the outputs, equation 2 is used again to get more accurate $x_b$. Finally, the predicted stress-strain curves can be described and plotted according to these parameters.

### *E. An Autoencoder Model for Initial Weights Extraction*

Due to the limited amount of labeled data (100 complete stress-strain curves), the MLP model may not yield very good results. Thus, in an attempt to improve the accuracy of our MLP model, we explore unsupervised learning methods with easy-to-obtain unlabeled data. Transfer learning (TL) techniques are widely used to solve the small dataset problem in machine learning. Recently, TL has gained significant interests in the machine learning and materials science community [20]. TL is used to develop a learning system where the knowledge learned from one domain (coined as source dataset) is used to improve the accuracy of another domain (coined as target dataset) which is generally small in size. This may be useful when the source problem has a lot more data than the target problem. There are two ways to perform TL, fine-tuning and feature extraction. Fine-tuning uses weights of the pre-trained model that is trained on the source problem as the starting

point for the training process and adapted in response to the target problem. This can also be called as TL performed via weight initialization scheme. The objective is to take advantage of data from the source problem to extract information that may be useful when learning or even when directly making predictions in the target problem.

Autoencoder is an unsupervised learning algorithm, which was first proposed by Rumelhartet et al. [21]. Autoencoder model can reconstruct its own data through training, the outputs of the model can realize the reproduction of inputs. The model consists of two main components: an encoder and a decoder. The encoder maps the original data to the hidden layer through the encoding function, and the decoder uses the decoding function to take the hidden layer as input to obtain the reconstruction of the data. The reconstructed data has the same structure as the original data. Autoencoder has many successful applications in many different domains, such as image classification, natural language processing, and material science [22], [23]. In order to make the reconstructed data be as close as possible to the original input during training, the autoencoder model needs to learn some potential information of the original data after training. Thus, the weights of layers after training contain the information of the training data, that could be transferred as initial weights for neural networks. Using weights learned from unlabeled data using unsupervised learning not only provides good initialization of network weights, but also improves the overall generalization performance [24].

Weight initialization based on autoencoder model is used in this study. The architecture of encoder network is the same as the MLP network. It consists of one input layer, one hidden layer, and one output layer. The number of neurons in the input layers depends on the input dimension. The hidden layer has 64 neurons and the output layer has 4 neurons. The decoder network is the opposite. The input layer has 4 neurons, the hidden layer has 64 neurons, and the number of neurons in the output layer is the same as the input layer of the encoder. We use unlabeled data from 8281 orientations as the training data to train the autoencoder model. Subsequently, the weights of the encoder were used to initialize the weights of the MLP model. The unlabeled data provides more orientation-based information to the model during the training process. We call the new MLP model with weight initialization as the Improved-MLP model.

## IV. Experiments And Results

### A. Evaluation Metrics

We evaluate the proposed framework by mean absolute error fraction $MAEf_0$, $MAEf_1$, $MAEf_2$ shown in equations 3, 4, 5. $MAEf_0$ compares the assumed curve and the actual curve, $MAEf_1$ compares the predicted curve and the assumed curve. $MAEf_2$ compares the predicted curve and the actual curve. $N$ is the number of steps for each stress-strain curve of an orientation. $Y\_actual_t$ is the t-th point's actual stress value, $Y\_assume_t$ is the assumed stress value of this point calculated by the assumed bilinear function and $Y\_pred_t$ is the predicted stress value of this point by the MLP model.

$$MAEf_0 = \frac{\sum_{t=0}^{N} \frac{|Y\_assume_t - Y\_actual_t|}{Y\_actual_t}}{N} \tag{3}$$

$$MAEf_1 = \frac{\sum_{t=0}^{N} \frac{|Y\_pred_t - Y\_assume_t|}{Y\_assume_t}}{N} \tag{4}$$

$$MAEf_2 = \frac{\sum_{t=0}^{N} \frac{|Y\_pred_t - Y\_actual_t|}{Y\_actual_t}}{N} \tag{5}$$

### B. Experiment Settings

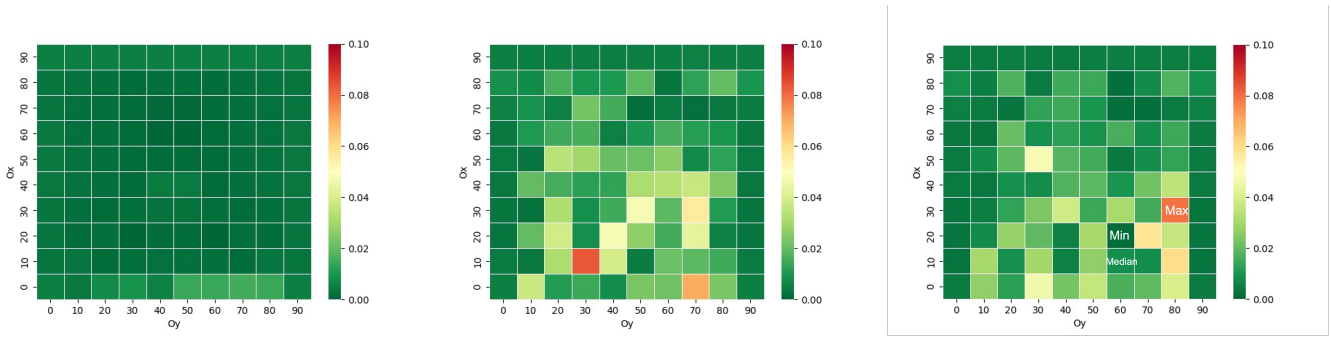
We use the leave-one-out cross-validation method to evaluate the proposed framework on the labeled dataset which has 100 data points. Several ML methods, such as Linear Regression, K-Nearest Neighbor algorithm (KNN), and Random Forest are used to compare with the MLP model. All machine learning models are implemented by scikit-learn using the default hyperparameters. The autoencoder model is implemented using Pytorch. The mean square error loss function is used with an Adam optimizer [25]. The learning rate is 0.0001. The number of epochs is 500. All experiments were conducted on a machine with 2.5 Ghz Intel(R) Xeon(R) Gold 6126 CPU.

### C. Results and Analysis

Figure 5(a) shows the $MAEf_0$ values of every orientation for 100 labeled data. The average value of $MAEf_0$ for 100 orientations is 0.0030, i.e., 0.30%.

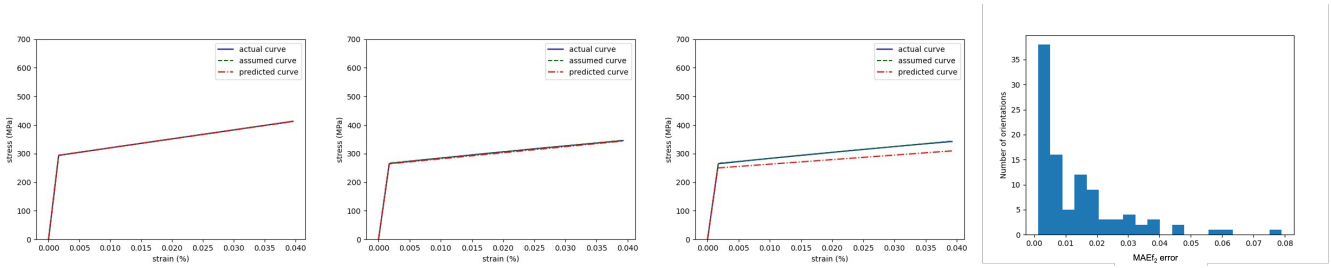
Tables I, II, III, IV, V show the results of different models using different combinations of $\mathbf{F}_1$, $\mathbf{F}_1^p$ and orientation number as inputs, respectively. The mse of parameters calculates the mean square error (MSE) of all four predicted parameters $m_1, m_2, c$ and $x_b$. The results in the tables are average mse of parameters, average $MAEf_1$, and average $MAEf_2$ from leave-one-out cross-validation. We can see that the results of two MLP models are better than other models for all combinations of inputs except when input is $\mathbf{F}_1^p$. The results of the Improved-MLP model are better than the MLP model for all combinations of inputs, which means using the information of larger unlabeled data can improve the performance of the MLP model trained on a small dataset. The results are better when input is $\mathbf{F}_1$ than when it is $\mathbf{F}_1^p$, which indicates $\mathbf{F}_1$ encodes more information of orientation. This insight is consistent with the knowledge that $\mathbf{F}_1$ includes deformation for both elastic and plastic parts while $\mathbf{F}_1^p$ represents just the plastic part of the stress-strain curve. The best result for $MAEf_2$ is 0.0141 (i.e., 1.41%) obtained by the Improved-MLP model when inputs are $\mathbf{F}_1$ and $\mathbf{F}_1^p$. The result demonstrates that the proposed Improved-MLP model can predict the stress-strain curves with high accuracy.

Figure 5(b)(c) shows the $MAEf_2$ values of every orientation for the MLP model and the Improved-MLP model,




(a) (b) (c)

Fig. 5: (a) $MAEf_0$ values; (b) $MAEf_2$ values of MLP model; (c) $MAEf_2$ values of Improved-MLP model for 100 orientations.




(a) (b) (c) (d)

Fig. 6: Actual stress-strain curve, assumed bilinear stress-strain curve and predicted bilinear stress-strain curve for orientations (a) 20-60-0, with minimum error $MAEf_2 = 0.0012$ ; (b) 10-60-0, with median error $MAEf_2 = 0.0086$; (c) 30-80-0, with maximum error $MAEf_2 = 0.0790$; (d) histogram of $MAEf_2$ error distribution.

TABLE I: Result comparison using $\mathbf{F}_1$ and $\mathbf{F}_1^p$ as inputs

| Models | average mse of parameters | average $MAEf_1$ | average $MAEf_2$ |
|---|---|---|---|
| LinearRegression | 0.5162 | 0.0819 | 0.0822 |
| KNeighborsRegressor | 0.5538 | 0.0517 | 0.0525 |
| RandomForestRegressor | 0.1551 | 0.0269 | 0.0278 |
| MLP | 0.0231 | 0.0141 | 0.0155 |
| **Improved-MLP** | **0.0213** | **0.0127** | **0.0141** |

TABLE II: Result comparison using $\mathbf{F}_1$, $\mathbf{F}_1^p$ and orientation number as inputs

| Models | average mse of parameters | average $MAEf_1$ | average $MAEf_2$ |
|---|---|---|---|
| LinearRegression | 0.5135 | 0.0819 | 0.0822 |
| KNeighborsRegressor | 0.5548 | 0.0517 | 0.0525 |
| RandomForestRegressor | 0.1469 | 0.0269 | 0.0278 |
| MLP | 0.0464 | 0.0222 | 0.0231 |
| **Improved-MLP** | **0.0351** | **0.0178** | **0.0186** |

TABLE III: Result comparison using $\mathbf{F}_1$ as inputs

| Models | average mse of parameters | average $MAEf_1$ | average $MAEf_2$ |
|---|---|---|---|
| LinearRegression | 0.4394 | 0.0753 | 0.0757 |
| KNeighborsRegressor | 0.4809 | 0.0581 | 0.0583 |
| RandomForestRegressor | 0.1892 | 0.0311 | 0.0318 |
| MLP | 0.0290 | 0.0151 | 0.0165 |
| **Improved-MLP** | **0.0278** | **0.0142** | **0.0156** |

respectively, when inputs are $\mathbf{F}_1$, $\mathbf{F}_1^p$. The average values of $MAEf_2$ in Figure 5[c] is lowest in all different models and all different combinations of inputs. We can see that the $MAEf_2$ values obtained by the Improved-MLP model are better than the MLP model for most orientations. This result demonstrates that the weight initialization based on autoencoder model can help the MLP model learn more information from a larger unlabeled dataset to get better performance.

Figure 6(a)(b)(c) show the actual, assumed, and predicted stress-strain curves of some example orientations obtained by the proposed framework, respectively. Figure 6(a) shows the stress-strain curve of orientation "20-60-0". The $MAEf_2$ value equals to 0.0012, which is the minimum error across

TABLE IV: Result comparison using $\mathbf{F}_1^p$ as inputs

| Models | average mse of parameters | average $MAEf_1$ | average $MAEf_2$ |
|---|---|---|---|
| LinearRegression | 0.5135 | 0.0819 | 0.0822 |
| KNeighborsRegressor | 0.5548 | 0.0517 | 0.0525 |
| **RandomForestRegressor** | **0.1386** | **0.0262** | **0.0271** |
| MLP | 1.0665 | 0.0961 | 0.0962 |
| Improved-MLP | 1.0665 | 0.0961 | 0.0962 |

TABLE V: Result comparison using orientation number as inputs

| Models | average mse of parameters | average $MAEf_1$ | average $MAEf_2$ |
|---|---|---|---|
| LinearRegression | 1.0262 | 0.0962 | 0.0960 |
| KNeighborsRegressor | 0.1646 | 0.0419 | 0.0421 |
| RandomForestRegressor | 0.1016 | 0.0344 | 0.0351 |
| MLP | 0.0792 | 0.0332 | 0.0336 |
| **Improved-MLP** | **0.0773** | **0.0331** | **0.0335** |

all orientations. Figure 6(b) shows the stress-strain curve of orientation "10-60-0". The $MAEf_2$ value equals to 0.0086, which is the median error across all orientations. Figure 6(c) shows the stress-strain curve of orientation "30-80-0". The $MAEf_2$ value equals to 0.0790, which is the maximum error across all orientations. Figure 6(d) shows the histogram of $MAEf_2$ error distribution. We can see that most $MAEf_2$ values are less than 0.002 (i.e., 2 %). These results demonstrate the proposed framework can predict the stress-strain with minimal error. Moreover, one of the biggest advantages of the proposed framework is the huge computational improvements over conventional simulation tools. These computational savings come from the fact that once an ANN model is trained and validated, the model does not need to run computationally expensive simulations to predict the data. Therefore, the proposed framework takes only a fraction of the time compared to numerical simulations. Generating a complete stress-strain curve of Ni-based superalloys for a given orientation takes around 2,400 s by CPFE simulations. The proposed framework takes $< 0.6$ s to predict the stress-strain curve, which means over 4000x speedup.

## V. Conclusion and Future Work

This paper presents an intelligent and effective AI/ML framework to predict the stress-strain curve of Ni-based superalloys. The proposed framework only needs a small number of labeled data (100 complete stress-strain curves) to train a MLP model. And transfer learning techniques based on autoencoder model are used to improve the accuracy of the MLP model with unlabeled data (initial $\mathbf{F}$ and $\mathbf{F}^p$ matrices). The results demonstrate that the proposed stress-strain curve prediction framework provides significant computational time improvements with minimal prediction error.

This work provides a viable solution to speed up the generation of stress-strain curves to save computational time and resources. In the future, we plan to extend this framework to more materials and build upon the proposed framework by integrating it with the CPFE simulation, in order to quickly calculate all the intermediate simulation variables, and not just the final stress-strain curve. This can potentially provide more interpretable information about the material's deformation behavior to the FEM practitioners, and also provide an opportunity to further improve the prediction accuracy of the model.

## Acknowledgment

This work is supported in part by the following grants: NIST award 70NANB19H005; DOE awards DE-SC0019358, DE-SC0021399; NSF award CMMI-2053929; and Northwestern Center for Nanocombinatorics.

## References

[1] A. Agrawal and A. Choudhary, "Perspective: Materials informatics and big data: Realization of the "fourth paradigm" of science in materials science," *Apl Materials*, vol. 4, no. 5, p. 053208, 2016.

[2] A. Agrawal and A. Choudhary, "Deep materials informatics: Applications of deep learning in materials science," *MRS Communications*, vol. 9, no. 3, pp. 779–792, 2019.

[3] D. Jha, V. Gupta, L. Ward, Z. Yang, C. Wolverton, I. Foster, W.-k. Liao, A. Choudhary, and A. Agrawal, "Enabling deeper learning on big data for materials informatics applications," *Scientific reports*, 2021.

[4] Y. Mao, Z. Yang, D. Jha, A. Paul, W.-k. Liao, A. Choudhary, and A. Agrawal, "Generative adversarial networks and mixture density networks-based inverse modeling for microstructural materials design," *Integrating Materials and Manufacturing Innovation*, pp. 1–11, 2022.

[5] D. Jha, V. Gupta, W.-k. Liao, A. Choudhary, and A. Agrawal, "Moving closer to experimental level materials property prediction using ai," *Scientific reports*, vol. 12, 2022.

[6] V. Gupta, W.-k. Liao, A. Choudhary, and A. Agrawal, "Brnet: Branched residual network for fast and accurate predictive modeling of materials properties," in *Proceedings of the 2022 SIAM International Conference on Data Mining (SDM)*, pp. 343–351, SIAM, 2022.

[7] V. Gupta, K. Choudhary, Y. Mao, K. Wang, F. Tavazza, C. Campbell, W.-k. Liao, A. Choudhary, and A. Agrawal, "Mppredictor: An artificial intelligence-driven web tool for composition-based material property prediction," *Journal of Chemical Information and Modeling*, 2023.

[8] Y. Mao, H. Lin, C. X. Yu, R. Frye, D. Beckett, K. Anderson, L. Jacquemetton, F. Carter, Z. Gao, W.-k. Liao, *et al.*, "A deep learning framework for layer-wise porosity prediction in metal powder bed fusion using thermal signatures," *Journal of Intelligent Manufacturing*, 2023.

[9] C. Yang, Y. Kim, S. Ryu, and G. X. Gu, "Prediction of composite microstructure stress-strain curves using convolutional neural networks," *Materials & Design*, vol. 189, p. 108509, 2020.

[10] A. Jenab, I. S. Sarraf, D. E. Green, T. Rahmaan, and M. J. Worswick, "The use of genetic algorithm and neural network to predict rate-dependent tensile flow behaviour of aa5182-o sheets," *Materials & Design*, vol. 94, pp. 262–273, 2016.

[11] U. Ali, W. Muhammad, A. Brahme, O. Skiba, and K. Inal, "Application of artificial neural networks in micromechanics for polycrystalline metals," *International Journal of Plasticity*, vol. 120, 2019.

[12] K. Koenuma, A. Yamanaka, I. Watanabe, and T. Kuwabara, "Estimation of texture-dependent stress-strain curve and r-value of aluminum alloy sheet using deep learning," *MATERIALS TRANSACTIONS*, 2020.

[13] A. Yamanaka, R. Kamijyo, K. Koenuma, I. Watanabe, and T. Kuwabara, "Deep neural network approach to estimate biaxial stress-strain curves of sheet metals," *Materials & Design*, vol. 195, p. 108970, 2020.

[14] D. Merayo, A. Rodríguez-Prieto, and A. M. Camacho, "Prediction of mechanical properties by artificial neural networks to characterize the plastic behavior of aluminum alloys," *Materials*, vol. 13, no. 22, 2020.

[15] S. G. Setti and R. Rao, "Artificial neural network approach for prediction of stress–strain curve of near $\beta$ titanium alloy," *Rare Metals*, 2014.

[16] S. Keshavarz and S. Ghosh, "Multi-scale crystal plasticity finite element model approach to modeling nickel-based superalloys," *Acta Materialia*, vol. 61, no. 17, pp. 6549–6561, 2013.

[17] S. Keshavarz, S. Ghosh, A. C. Reid, and S. A. Langer, "A non-schmid crystal plasticity finite element approach to multi-scale modeling of nickel-based superalloys," *Acta Materialia*, vol. 114, pp. 106–115, 2016.

[18] F. Roters, P. Eisenlohr, L. Hantcherli, D. D. Tjahjanto, T. R. Bieler, and D. Raabe, "Overview of constitutive laws, kinematics, homogenization and multiscale methods in crystal plasticity finite-element modeling: Theory, experiments, applications," *Acta Materialia*, 2010.

[19] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[20] V. Gupta, K. Choudhary, F. Tavazza, C. Campbell, W.-k. Liao, A. Choudhary, and A. Agrawal, "Cross-property deep transfer learning framework for enhanced predictive analytics on small materials data," *Nature communications*, vol. 12, no. 1, pp. 1–10, 2021.

[21] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, 1986.

[22] A. Ashfahani, M. Pratama, E. Lughofer, and Y.-S. Ong, "Devdan: Deep evolving denoising autoencoder," *Neurocomputing*, vol. 390, 2020.

[23] X. He, Q. He, and J.-S. Chen, "Deep autoencoders for physics-constrained data-driven nonlinear materials modeling," *Computer Methods in Applied Mechanics and Engineering*, vol. 385, 2021.

[24] Y. Bengio, *Learning deep architectures for AI*. Now Publishers Inc, 2009.

[25] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.