A deep learning framework for time-series processing-microstructure-property prediction

Yuwei Mao*, Mahmudul Hasan†, Claire Lee*, Muhammed Nur Talha Kilic‡, Vishu Gupta*, Wei-keng Liao*, Alok Choudhary*, Pinar Acar†, Ankit Agrawal*
*ECE Department, ‡CS Department, Northwestern University, Evanston, USA
{yuweimao2019,csl, talha.kilic, vishugupta2020}@u.northwestern.edu, {wkliao,choudhar,ankitag}@eecs.northwestern.edu
†Virginia Tech, USA, {mdmahmudul, pacar}@nist.gov

***Abstract*—A process simulator provides valuable insights into the evolution of microstructures under various elementary processes, employing the Orientation Distribution Function (ODF) as a representation of the microstructure's texture. However, such simulations often involve complex physical computations, making them time-consuming. To address this, our study introduces an artificial intelligence (AI)-based framework to predict the microstructural texture of polycrystalline materials using a specified deformation process. As a case study, we apply our framework to copper. The dataset includes 3,125 unique processing parameter combinations and their corresponding ODF vectors generated using a process simulator. The resulting predictions enable the calculation of homogenized properties. As opposed to traditional material processing simulations, our AI-driven framework offers faster results with minimal error rates (less than 0.5%). This indicates that our approach is a promising tool for rapidly predicting processing-specific microstructures and properties, thereby offering significant improvements over conventional simulation techniques.**

***Index Terms*—seq2seq, LSTM, time-series prediction**

I. Introduction

Polycrystalline materials play a vital role in a wide range of industrial applications due to their versatile properties and cost-effectiveness. These materials exhibit complex microstructures, which significantly impact their mechanical and physical properties. Understanding the relationship between processing, structure, property and performance (PSPP) in polycrystalline materials is crucial for the development of advanced materials with tailored properties, enabling better performance in various applications such as aerospace, automotive, and electronics industries [1]–[3].

However, traditional experimental approaches that explore the optimum processing routes for a specific texture are often laborious, time-consuming, and expensive, as they rely heavily on trial and error. In recent years, computational methods have emerged as a promising alternative to replace experimental approaches, expediting the design of microstructures with desired textures. The fast and accurate prediction of texture evolution during processing holds the key to bridging the gap between material design and manufacturing efforts for polycrystalline materials [4], [5].

A process simulator of microstructure evolution can be employed to learn the evolution of microstructures under each elementary process, utilizing the ODF as a means to represent microstructure texture. The ODF describes the volume density of crystals of different orientations in a microstructure, allowing for a comprehensive representation of the material's texture. Given an arbitrary initial texture (ODF) and a set of processing parameters, the future sequential ODFs can be predicted using the process simulator. However, the simulation is very time-consuming because of complex physical calculations.

In recent years, AI and ML techniques have emerged as promising tools for accelerating the understanding of material behavior and guiding the development of new materials with desired properties. These techniques have demonstrated success in various material science applications, such as predicting crystal structures, estimating mechanical properties, and simulating deformation processes. Several machine-learning approaches have been employed to model the microstructural evolution in different materials, such as metals, alloys, and composites [6], [7].

In this study, our objective is to predict the ODF, and subsequently, property prediction is executed by utilizing processing parameters along with the initial ODFs. If successful, the resulting model has the potential to replace the process simulator, thereby significantly diminishing running time. However, there are some challenges: 1. The dimensions of ODF vectors exhibit non-linearity with respect to time steps. 2. Certain dimensions of the ODF vectors initially show linearity but deviate towards non-linearity in future steps. 3. Our objective is to minimize the number of historical ODFs used to achieve reduction in running time. However, this approach may pose challenges in obtaining accurate predictions.

To address these challenges, we propose an artificial intelligence-based framework that employs an encoder-decoder model comprising Long Short-Term Memory (LSTM) layers. The model is specifically designed to predict alterations in the ODF of polycrystalline materials during a given deformation process. The ODF is a statistical representation of the crystallographic orientations within a polycrystalline material, which directly affects its overall properties. Our study specifically uses copper as the subject material due to its excellent electrical and thermal conductivity, high ductility, and resistance to corrosion. Its robust mechanical properties also make it a suitable choice for various applications across industries such as electronics, telecommunications, power gen-

eration, and building construction [8]. The dataset used for this study is generated from material processing simulations, comprising 3,125 unique combinations of processing parameters. Each combination yields a sequence of ten ODF vectors as the output of the processing. Our proposed framework employs the initial ODF and processing parameters as input data, and predicts the evolution of texture in terms of ODFs for the given deformation process. Subsequently, homogenized properties are calculated using the predicted ODFs, providing a comprehensive prediction of the material's behavior under these processing conditions.

The proposed AI-driven framework provides a fast, cost-effective, and accurate method for predicting texture evolution in polycrystalline materials, ultimately paving the way for optimized microstructures with tailored properties, outperforming the material processing simulation with only a minimal compromise in accuracy.

II. Deformation Process Modeling with ODF Approach

Polycrystalline materials are composed of multiple crystals with various crystallographic orientations, which determine the microstructural texture. This texture is mathematically described by the ODF, which allows for efficient deformation process modeling compared to computationally expensive finite element solvers. The ODF, represented by $\mathbf{A}(\mathbf{r}, t)$, indicates the volume density of the crystals in the orientation space, $\mathbf{r}$, at a specific time t.

The Taylor approximation [9] may be used to calculate the volume-averaged (homogenized) properties of polycrystalline materials utilizing their single crystal properties and microstructural orientation information. The following equation can be used to calculate the volume-averaged elastic characteristics $\mathbf{C}^{avg}$ of polycrystalline metals.

$$\mathbf{C}^{avg} = < \mathbf{C} > \tag{1}$$

Here, $\mathbf{C}$ is the stiffness tensor of each crystal and $< . >$ is the symbol of averaging. Similarly, if any property of a single crystal $\chi(r)$ is dependent on the crystal orientation, then the homogenized polycrystal property $< \chi >$ can be determined by performing the averaging over the ODF, that could be written as equation 2:

$$< \chi > = \int_{\Omega} \chi(r)\mathbf{A}(r)\, dv \tag{2}$$

Here $\chi(r)$ represents the single-crystal material properties. After the crystallographic symmetries (hexagonal, cubic) are enforced, the integration is conducted in the fundamental region, Ω, of the orientation space.

During the deformation process, the ODFs change due to the reorientation of the grains. They evolve from the initial ODFs (at time $t = 0$) to the final deformed ODFs (at time $t = t_{final}$). Each deformation process, such as tension/compression and shear, generates a particular ODF as output after applying a load for a specific amount of time.

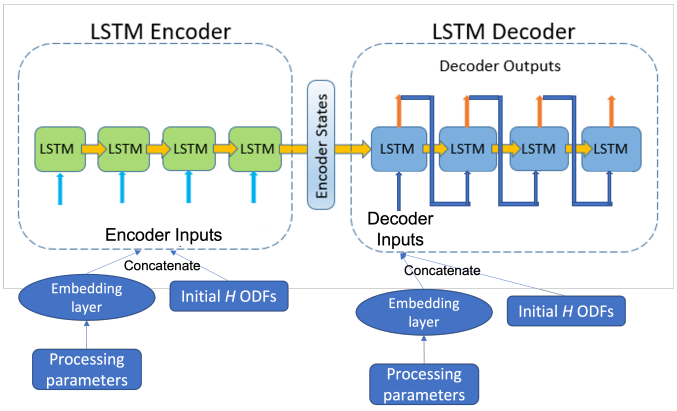

Fig. 1. Encoder-decoder LSTM architecture.

III. Methods

A. Model Architecture

The encoder-decoder model is a type of neural network architecture that has been widely used in various fields, including natural language processing and computer vision. It is a sequence-to-sequence model that consists of two main components: an encoder and a decoder. The encoder takes an input sequence and maps it to a fixed-length vector representation, which captures the most relevant information from the input sequence. This vector representation is then passed to the decoder, which generates an output sequence based on the encoded information.

One type of neural network layer commonly used for time series data in the encoder-decoder model is LSTM layer. LSTMs are a type of recurrent neural network (RNN) that is designed to handle sequential data by allowing information to persist over time. The proposed model is an encoder-decoder architecture that uses LSTM layers for both encoding and decoding. Figure 1 shows the architecture of the proposed encoder-decoder LSTM model. The encoder takes H ODF vectors and processing parameters as input. The processing parameters are embedded using a linear layer, and the resulting feature vector is concatenated to the ODF vectors. The hidden layer of the encoder is fed into the decoder, which generates a sequence of future ODF vectors. The decoder takes the initial H ODF vectors and processing parameters combination as input.

Teacher-forcing is a technique used in sequence-to-sequence models, such as the encoder-decoder model, during training to improve the accuracy of the generated output sequence. In this work, teacher-forcing is used to provide the correct output from the previous time step as input with probability *P*. During testing, the decoder generates its own output at each step based on the previous output and processing parameters.

IV. Results

A. Experimental settings

The model was trained using the Adam optimizer and a mean squared error (MSE) loss function. The learning rate

was set to 0.001, and the batch size was set to 32. We divide the training set into the training set and a validation set with proportions of 70% and 30%, respectively. We select the best model in the validation set as the model to predict the results of the test set. The performance of the model was evaluated on the test set. We compared the performance of our proposed model with that of several baselines, including a linear regression model, a ridge model, and a feedforward neural network.

B. Dataset

We generated a dataset for this study using a strain-rate independent crystal plasticity simulation. The deformation simulation can be performed for tension/compression and shear forces in XY, YZ, and XZ planes. The strain rate is varied for each deformation process. In order to cover a wider range of real data distribution, we generate process data by taking the strain rate values from a list of (0, 0.25, 0.5, 0.75, 1) for each process, resulting in $5^5 = 3,125$ (five options for each parameter, and a total of five parameters) unique combinations. For every combination of processing parameters, the simulator provides the final deformed ODF with nine intermediate steps of ODF evolution. Therefore, we obtain a sequence of ten ODF vectors ($ODF_1, ODF_2, ODF_3, ..., ODF_{10}$) as output, all starting from the same initial ODF vector. Each ODF is represented by a 76-dimensional vector. We divided the dataset, consisting of ODF vectors and their corresponding processing parameters, into training and test sets, with proportions of 70% and 30%, respectively. This division ensured that our model was trained and tested on independent and representative data samples.

C. Evaluation metrics

We employed the ODF_ave_ave_mape metric to compare predicted ODFs. First, we calculated the mean absolute percentage error (mape) for each predicted ODF of each processing combination. Then, we computed the average mape value of F future ODFs as the ave_mape of each processing combination. Finally, we determined the average ave_mape of all testing processing combinations to obtain the ODF_ave_ave_mape, as shown in Eq. (3). P is the number of processing combinations in the test set. F is the number of predicted ODFs for each processing combination. N is the number of dimensions for each ODF vector. This metric allowed us to compare the ODF prediction performance of different models, as demonstrated in the "ODF mape" column in Tables I and II.

$$ODF\_ave\_ave\_mape = \frac{1}{P}\sum_{process\_i}^{P}\frac{1}{F}\sum_{step\_j=1}^{F}\frac{1}{N}\sum_{k=1}^{N}\left|\frac{ODF\_actual_{step\_j,k}^{process\_i} - ODF\_predicted_{step\_j,k}^{process\_i}}{ODF\_actual_{step\_j,k}^{process\_i}}\right| \times 100\% \tag{3}$$

After obtaining the predicted ODFs, we derived property matrices C and S using homogenization. We designated these as predicted C and predicted S. As these matrices were not predicted directly by the models but were calculated using homogenization, they had fewer errors compared with predicting property directly. We adopted a similar method to evaluate the performance of predicted properties. For instance, for each processing combination, we calculated the mape of every element of F predicted C matrices. Subsequently, we computed the average mape of all processing combinations. In this manner, we obtained a 6×6 matrix representing the average mape of every element of predicted C. These results are shown in the "Property mape" column in Tables I and II. We determined the average mape of predicted S matrices using the same method.

Meanwhile, we utilize the selected_property to assess the predicted C matrix using a single value. The selected_property is computed from the chosen elements (which comprise $C_{11}, C_{22}, C_{33}, C_{44}, C_{55}, C_{66}, C_{21}, C_{31}, C_{32}$, resulting in a total of nine elements). For each predicted C matrix corresponding to a specific processing combination, we can determine the mean absolute percentage error (mape) of the selected elements as a representation of the error in that predicted C matrix. Subsequently, we calculate the average mape value for F predicted C matrices as the average_mape for that particular processing combination. Lastly, we compute the average of the average_mape values for all testing processing combinations to obtain the C_ave_ave_mape of the predicted C matrices, which can be expressed as Eq. (4).

$$C\_ave\_ave\_mape = \frac{1}{P}\sum_{process\_i}^{P}\frac{1}{F}\sum_{step\_j=1}^{F}\frac{1}{N}\sum_{k=1}^{N}\left|\frac{C\_actual_{step\_j,k}^{process\_i} - C\_predicted_{step\_j,k}^{process\_i}}{C\_actual_{step\_j,k}^{process\_i}}\right| \times 100\% \tag{4}$$

We can derive the S_ave_ave_mape using an analogous method. Both C_ave_ave_mape and S_ave_ave_mape assist in comparing the performance of different models in predicting property matrices. These values can be found under the "Selected_property mape" column in tables I and II.

D. Evaluation on ten predicted ODFs using only initial ODF ($F = 10$)

We employed the Linear Regression model, Ridge model, and neural networks to predict ten ODFs using only the initial ODF and corresponding processing parameters. Consequently, the input feature dimension is 81 (76+5). During both the training and testing phases, we normalized the input features to ensure a unit norm. The results can be found in Table I. Bold numbers in the Property mape column have been selected as selected_property for calculating the Selected_property mape. It is evident that the neural network (NN) model outperforms the others in terms of ODF mape and Selected_property MAPE for both C and S matrices, with an ODF MAPE of 8.22%. Therefore, we utilize the NN model as a baseline for

TABLE I
RESULTS OF DIFFERENT MODELS WHEN $H = 1$

Models	ODF mape	Property	Property mape						Selected_property mape
LR	102.50%	C	**0.89%**	0.78%	0.83%	527.48%	726.88%	5419.88%	2.32%
			0.78%	**1.79%**	4.12%	230.74%	3893.89%	12950.31%	
			0.83%	**4.12%**	**0.85%**	546.16%	1494.96%	7176.23%	
			527.48%	230.74%	546.16%	**7.76%**	5023.89%	745.31%	
			726.88%	3893.89%	1494.96%	5023.89%	**1.37%**	2421.54%	
			5419.88%	12950.31%	7176.23%	745.31%	2421.54%	**2.53%**	
		S	**1.40%**	2.36%	9.29%	744.98%	747.38%	131125.80%	5.50%
			2.36%	**5.30%**	16.47%	140.99%	1720.90%	7611.16%	
			9.29%	**16.47%**	**2.38%**	938.04%	1204.08%	6345.09%	
			744.98%	140.99%	938.04%	**8.56%**	11256.34%	963.66%	
			747.38%	1720.90%	1204.08%	11256.34%	**1.47%**	9467.69%	
			131125.80%	7611.16%	6345.09%	963.66%	9467.69%	**2.27%**	
Ridge	19.26%	C	**0.47%**	0.45%	0.48%	295.69%	248.81%	240.23%	0.51%
			0.45%	**0.48%**	0.47%	282.01%	296.96%	526.97%	
			0.48%	**0.47%**	**0.48%**	251.72%	954.86%	317.66%	
			295.69%	282.01%	251.72%	**0.58%**	258.52%	424.72%	
			248.81%	296.96%	954.86%	258.52%	**0.62%**	696.11%	
			240.23%	526.97%	317.66%	424.72%	696.11%	**0.55%**	
		S	**0.57%**	0.54%	1.27%	446.24%	203.76%	1004.66%	0.73%
			0.54%	**0.61%**	1.27%	257.95%	586.39%	501.38%	
			1.27%	**1.27%**	**0.57%**	419.62%	320.58%	275.37%	
			446.24%	257.95%	419.62%	**0.59%**	390.46%	555.31%	
			203.76%	586.39%	320.58%	390.46%	**0.62%**	17621.94%	
			1004.66%	501.38%	275.37%	555.31%	17621.94%	**0.55%**	
NN	8.22%	C	**0.46%**	0.46%	0.44%	115.88%	109.18%	117.65%	0.45%
			0.46%	**0.49%**	0.40%	130.41%	157.29%	318.47%	
			0.44%	**0.40%**	**0.49%**	132.68%	433.37%	182.02%	
			115.88%	130.41%	132.68%	**0.35%**	165.00%	221.43%	
			109.18%	157.29%	433.37%	165.00%	**0.39%**	340.09%	
			117.65%	318.47%	182.02%	221.43%	340.09%	**0.55%**	
		S	**0.48%**	0.48%	0.63%	187.93%	95.63%	1631.53%	0.53%
			0.48%	**0.54%**	0.81%	131.36%	195.86%	284.43%	
			0.63%	**0.81%**	**0.55%**	182.36%	207.15%	161.45%	
			187.93%	131.36%	182.36%	**0.35%**	201.34%	255.03%	
			95.63%	195.86%	207.15%	201.34%	**0.39%**	9231.95%	
			1631.53%	284.43%	161.45%	255.03%	9231.95%	**0.56%**	

TABLE II
RESULTS OF DIFFERENT MODELS WHEN $H = 2$

Models	ODF mape	Property	Property mape						Selected_property mape
NN	8.01%	C	**0.39%**	0.47%	0.57%	221.58%	92.15%	185.68%	0.48%
			0.47%	**0.46%**	0.43%	135.06%	78.33%	321.44%	
			0.57%	**0.43%**	**0.41%**	78.56%	347.79%	283.54%	
			221.58%	135.06%	78.56%	**0.51%**	180.58%	298.92%	
			92.15%	78.33%	347.79%	180.58%	**0.69%**	118.33%	
			185.68%	321.44%	283.54%	298.92%	118.33%	**0.38%**	
		S	**0.32%**	0.37%	0.52%	249.62%	76.76%	3472.73%	0.48%
			0.37%	**0.47%**	0.70%	166.15%	195.13%	381.32%	
			0.52%	**0.70%**	**0.35%**	85.41%	114.02%	218.15%	
			249.62%	166.15%	85.41%	**0.51%**	161.60%	458.39%	
			76.76%	195.13%	114.02%	161.60%	**0.70%**	5852.60%	
			3472.73%	381.32%	218.15%	458.39%	5852.60%	**0.38%**	
seq2seq	3.83%	C	**0.46%**	0.46%	0.45%	71.67%	55.88%	93.37%	0.46%
			0.46%	**0.45%**	0.46%	111.08%	143.05%	167.49%	
			0.45%	**0.46%**	**0.46%**	72.34%	193.98%	90.01%	
			71.67%	111.08%	72.34%	**0.46%**	65.67%	102.44%	
			55.88%	143.05%	193.98%	65.67%	**0.45%**	161.21%	
			93.37%	167.49%	90.01%	102.44%	161.21%	**0.46%**	
		S	**0.46%**	0.45%	0.51%	121.66%	44.36%	740.03%	0.46%
			0.45%	**0.45%**	0.46%	67.47%	132.55%	120.89%	
			0.51%	**0.46%**	**0.46%**	93.17%	115.11%	77.98%	
			121.66%	67.47%	93.17%	**0.46%**	104.90%	134.99%	
			44.36%	132.55%	115.11%	104.90%	**0.45%**	933.40%	
			740.03%	120.89%	77.98%	134.99%	933.40%	**0.46%**	

comparison with the proposed seq2seq model in subsequent experiments.

E. Evaluation on nine predicted ODFs using two initial ODFs ($F = 9$)

In Table II, we compare the results of the neural network (NN) model and seq2seq model using $H = 2$ and the processing parameters as inputs. As a result, the input feature dimension is 157 ($76 \times 2 + 5$). We can observe that the seq2seq model achieves a significantly lower ODF mape than the NN model (3.82% vs. 8.01%) and a lower Selected_property mape for both C and S matrices ((0.46% vs. 0.48%)). This demonstrates that the proposed seq2seq model can achieve better results by learning the time-series relationship, capturing not only the values of future ODFs but also the trend of every dimension in the sequence of ODFs.

V. DISCUSSION

In this study, we have presented a novel approach for predicting the sequence of deformed ODFs and their corresponding property matrices (C and S) after load application based on the seq2seq model with LSTM. Our results demonstrate the effectiveness of the proposed seq2seq model in capturing complex trends and predicting accurate values. The results of our experiments show that the seq2seq model consistently outperforms traditional machine learning models and the NN model in predicting ODFs and property matrices. This superior performance is attributed to the ability of the LSTM seq2seq model to learn and capture the time-series relationship in the input data. These findings have important implications for materials science and engineering, where accurate predictions of microstructure and properties are crucial for material design and optimization.

Despite the promising results, our study has several limitations. Firstly, our study focused on a specific set of materials and processing conditions. The generalizability of our approach to other materials and processing conditions needs to be further investigated. Moreover, the performance of the seq2seq model with different types of input features, such as crystallographic information or process parameters, could be explored to enhance prediction accuracy.

ACKNOWLEDGMENT

This work was supported primarily by National Science Foundation (NSF) CMMI awards 2053929/2053840. Partial support from NIST award 70NANB19H005, DOE award DE-SC0021399, and Northwestern Center for Nanocombinatorics is also acknowledged.

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, P. D. Deshpande, A. Cecen, G. P. Basavarsu, A. N. Choudhary, and S. R. Kalidindi, "Exploration of data science techniques to predict fatigue strength of steel from composition and processing parameters," *Integrating Materials and Manufacturing Innovation*, vol. 3, no. 1, pp. 90–108, 2014.

[3] 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.

[4] D. Sarkar, B. S. Reddy, and B. Basu, "Implementing statistical modeling approach towards development of ultrafine grained bioceramics: Case of zro2-toughened al2o3," *Journal of the American Ceramic Society*, vol. 101, no. 3, pp. 1333–1343, 2018.

[5] G. Tapia, L. Johnson, B. Franco, K. Karayagiz, J. Ma, R. Arroyave, I. Karaman, and A. Elwany, "Bayesian calibration and uncertainty quantification for a physics-based precipitation model of nickel–titanium shape-memory alloys," *Journal of Manufacturing Science and Engineering*, vol. 139, no. 7, 2017.

[6] A. A. K. Farizhandi and M. Mamivand, "Spatiotemporal prediction of microstructure evolution with predictive recurrent neural network," *Computational Materials Science*, vol. 223, p. 112110, 2023.

[7] D. Montes de Oca Zapiain, J. A. Stewart, and R. Dingreville, "Accelerating phase-field-based microstructure evolution predictions via surrogate models trained by machine learning methods," *npj Computational Materials*, vol. 7, no. 1, p. 3, 2021.

[8] N. J. Robinson and D. R. Winge, "Copper metallochaperones," *Annual review of biochemistry*, vol. 79, pp. 537–562, 2010.

[9] G. I. Taylor, "Plastic strain in metals," *J. Inst. Metals*, vol. 62, 1938.