Solid Freeform Fabrication 2022: Proceedings of the 33rd Annual International Solid Freeform Fabrication Symposium – An Additive Manufacturing Conference
Reviewed Paper

TRANSFERRING PROCESS MAPPING KNOWLEDGE ACROSS SS316L AND IN718 IN LASER DIRECTED ENERGY DEPOSITION USING MACHINE LEARNING

Nandana Menon[1], Sudeepta Mondal[1, 2], and Amrita Basak[1, *]

[1]Department of Mechanical Engineering, Pennsylvania State University, University Park, PA 16802

[2]Department of Mathematics, Pennsylvania State University, University Park, PA 16802

* Corresponding Author: aub1526@psu.edu

Abstract

Laser-directed energy deposition additive manufacturing processes have several parameters that impact the melt pool properties, which in turn affect the microstructure of the part. Computational investigations are regularly implemented; however, these investigations must be repeated for each material of interest. In this paper, a transfer learning approach is proposed to address this challenge. Using an analytical model, input-output data pairs are generated for a nickel-based alloy, IN718, and an iron-based alloy, SS316L. A baseline neural network is trained for SS316L. The capability of transfer learning is analyzed with parametric retraining of the percentage of data used and the number of retrained layers of the SS316L baseline network on the IN718 data. With just 10% data and one hidden layer retrained, accuracies above 90% are observed. The results show that the acquired printability knowledge can be transferred across material systems without requiring a significant amount of data for a new material system.

Introduction

Laser-directed energy deposition (L-DED) is driven by numerous controllable process parameters like laser power, scan velocity (traverse speed), laser scanning pattern, hatch spacing, feedstock feed rate, etc. Although all these parameters must be optimized to obtain the desired characteristics, laser power and scan velocity are of primary interest, as they offer a wide window for modifications and improvements in L-DED systems [1]. Much of the L-DED research has focused primarily on finding useful combinations of process parameters to ensure uniform deposit [2]. An important process characteristic that affects the deposit quality is the melt pool morphology. A uniform melt pool size correlates to a homogeneous microstructure. Hence, there is significant interest in understanding the relationship between process parameters and melt pool size. The effect of the process parameters on the melt pool size can be understood and expanded over the entire processing space by an approach patented as process mapping [3]. Process mapping techniques map process characteristics as a function of primary process parameters based on simulation or experimental results [4].

Traditionally, process maps are developed by design-of-experiment (DoE) which usually involves trial and error, and is time-consuming and costly. Modeling and simulation are relatively inexpensive means to aid in trial-and-error-based experimental investigations. Johnson et al. [5] used

analytical and thermophysical models to develop process maps linking laser parameters to non-dimensional melt pool parameters. Beuth et al. [6], [7] used a combination of results from experimental and numerical models to develop process maps connecting the steady-state as well as transient melt pool size to process parameters. In recent years, machine learning (ML) has been increasingly employed in additive manufacturing due to its unprecedented performance in data prediction tasks. Extreme gradient boosting (XGBoost) and long short-term memory (LSTM) were used by Zhang et al. [8] to build the ML models for predicting the melt pool temperature. Caiazzo et al. [9] used an artificial neural network to estimate the process parameters required to obtain a specific melt pool geometry. Mondal et al. [10] used Gaussian Process-based regression to predict the melt pool depth for the nickel superalloy CMSX-4®. This study was extended by Menon et al. [11] by employing a multi-fidelity Gaussian Process via co-kriging two independent thermal models of hierarchical fidelities to accelerate the prediction of melt pool depth with uncertainty quantification. A similar use of multi-fidelity models for melt pool modeling was performed by Huang et al. [12] where transfer learning was used to combine feature representations of the two fidelity levels to efficiently model the melt pool in electron beam additive manufacturing of Ti-6Al-4V.

Overall, there are significant evidences that, given a series of reliable training data of the property of interest (output) at different combinations of process parameters (input), a process map can be generated by these discrete data points using various ML regression techniques. However, a major drawback of process maps generated by these traditional DoE and conventional ML techniques is the need to remodel and generate new data for every new material system. Convention ML techniques are designed to work in isolation i.e., training and test data are drawn from the same feature space and the same distribution [13]. Collecting new data and rebuilding the model can often be an expensive or impossible task in most real-world applications. At present, there is limited knowledge on how to transfer the "process parameters – melt pool property" relationship knowledge across metal alloy systems and processes.

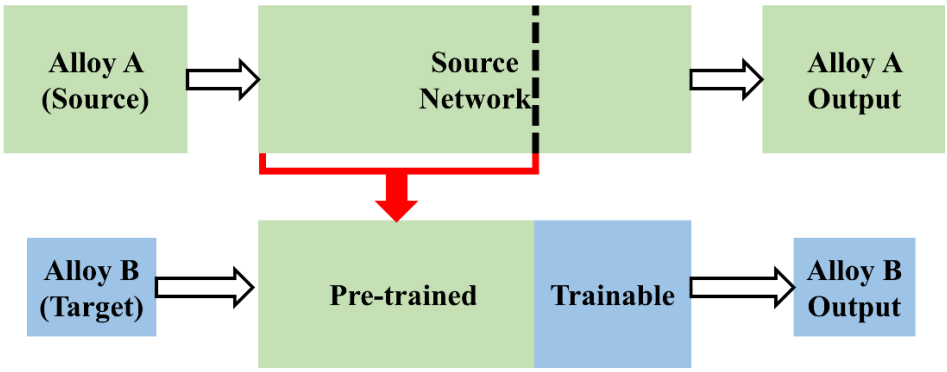

Figure 1. The framework of Transfer Learning. Note that the data for Alloy B (Target) is smaller than the data for Alloy A (Source).

In this paper, the authors attempt to address this research gap using Transfer learning (TL). TL is a machine learning framework implemented to overcome this isolated learning paradigm. It involves reusing models trained on a source task for a target task, thereby essentially transferring knowledge between the domains, as shown in Figure 1. TL is especially beneficial in cases where limited training data is available and learning a model from scratch can be ineffective. TL thus provides a solution to the lack of understanding of how the process mapping knowledge from one

alloy system can be transferred and applied to a different alloy system. The AM problem being addressed herein is the transfer of knowledge of process mapping for one alloy system (source), for which abundant data is available, to predict process maps for another alloy system (target), for which data available is limited, and the generation of new data is expensive. The nickel-based superalloy, IN718, is selected as the target alloy while the relatively inexpensive SS316L alloy is chosen as the source alloy. An analytical model, developed by Eagar and Tsai, is validated for SS316L and IN718 independently. Both models are then used to generate the input-output pairs. Thereafter, using transfer learning, a neural network trained on the SS316L data is extended to predict the "process parameters – melt pool property" relationship for IN718.

This article is organized as follows: the current section is followed by the methods section that outlines the data generation, neural network development, and implementation of TL. This is followed by the results and discussion that delineate the performance and justify the efficacy of TL.

Methods

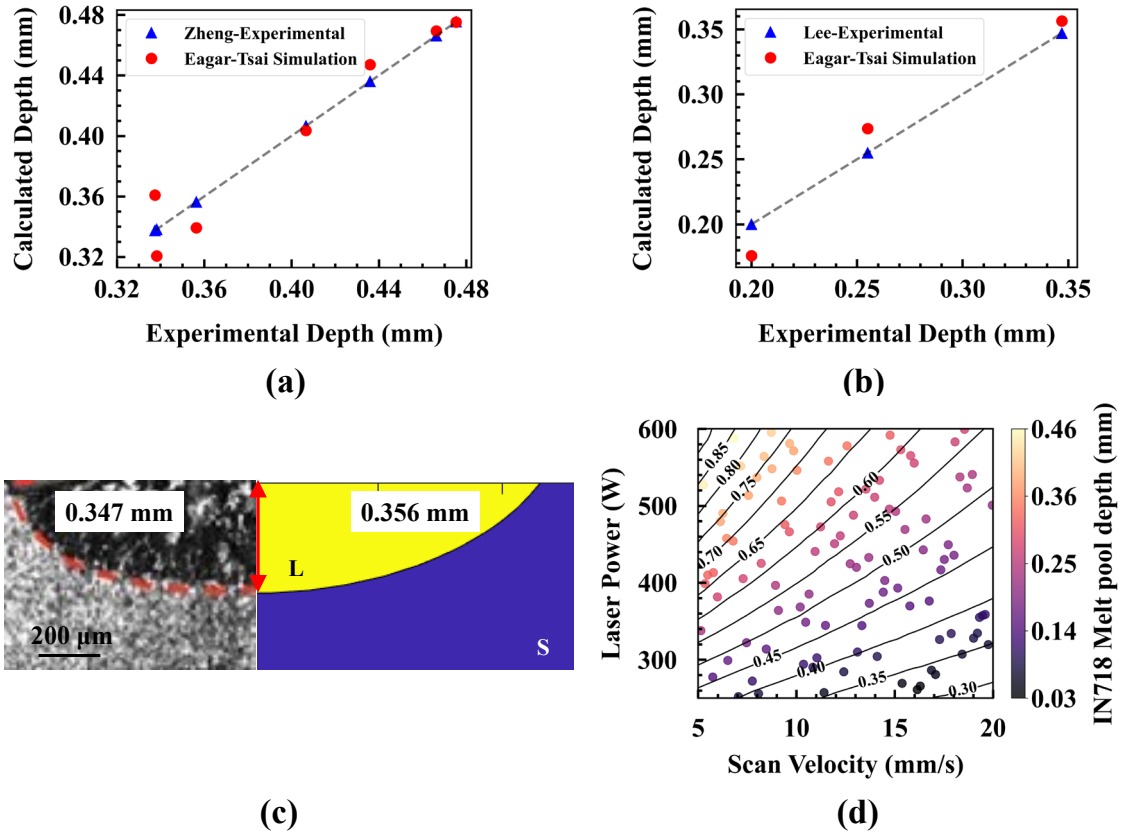

Figure 2. Comparison between Eagar-Tsai predictions and experimental observations for L-DED of (a) SS316L and (b) IN718. (c) Comparison of the experimental and calculated melt pool dimensions for IN718 where 'L' is the liquidus zone and 'S' is the solidus zone. (d) Melt pool depth of IN718 superimposed on a process map of melt pool depth for SS316L, over the same process parameter space. The black curves correspond to the contours of melt pool depth for SS316L while the markers correspond to the melt pool depth of IN718 color coded as per the secondary Y-axis.

Analytical Thermal Modeling of L-DED

The Eagar-Tsai [14] analytical model is a thermal model that is still popularly employed in the metal-AM community due to its ease of use and simplicity. It predicts the three-dimensional temperature distribution produced by a moving Gaussian heat source over a semi-infinite plate. The following assumptions make the model computationally inexpensive:

1. Absence of convective and radiative heat transfer.
2. Constant average thermal properties.
3. Quasi-steady state semi-infinite medium.
4. Absence of phase change and Marangoni flow.

Despite the simplifications, the model can be tuned to encompass most of the experimental scatter. Two experimentally validated models are generated for the source material, SS316L, and the target material, IN718. Once the temperature field is solved, the liquidus temperatures are used to identify the 3D iso-surface where the temperature equals the liquidus temperature of the material of interest. This iso-surface represents the melt pool boundary from which the melt pool dimensions can be extracted. The thermophysical properties of SS316L and IN718 are provided in Table 1. Figure 2 (a) and (b) show the predictions obtained from the Eagar-Tsai model with the experimental SS316L [15] and IN718 [16] data, respectively, demonstrating good agreement. A representative image of the melt pool for IN718 is shown in Figure 2 (c).

Table 1. Thermophysical properties of SS316L and IN718.

	SS316L	**IN718**
Density (kg/m^3)	7990	8190
Specific Heat (J / (kgK))	500	577
Thermal conductivity (W/(mK))	16.3	27.92
Liquidus Temperature (K)	1658	1623

Since Eagar-Tsai is computationally inexpensive, the process parameter space is densely sampled using the Latin Hypercube Sampling (LHS)-based DoE method and evaluated for the steady-state melt pool depth. LHS is implemented to select input points from the process parameter space consisting of P ranging from 250 W to 600 W and v ranging from 5 mm/s to 20 mm/s. The Eagar-Tsai model is interrogated with SS316L thermophysical properties to generate the output data (i.e., melt pool depth) for 200 input points. Similarly, the Eagar Tsai model is interrogated with IN718 thermophysical properties to generate the output data (i.e., melt pool depth) for IN718 for just 20 input points from another LHS. The choice of material and their respective number of data points are selected to reflect the practical limitations involved in process an expensive alloy such as IN718. The difference between the melt pool depths for the two alloys is highlighted by superimposing the melt pool depths of IN718 over the process map of SS316L for the predefined process parameter range in Figure 2 (d).

Development of the Source Neural Network

An artificial neural network (NN) consists of layers of interconnected nodes of weights with associated weights and biases that are optimized over a training process [17]. Neural network functions are generally written in linear algebraic form: $z^L = \sigma(W^L z^{L-1} + b^L)$. Here, z^L is the activation at layer L, W^L is the weight matrix connecting layer L and layer $L-1$, b^L is the bias at each layer L, and σ is the activation function. The neural networks, in this study, are developed using Keras [18], a deep learning API written in Python that is running on top of the machine learning platform, TensorFlow.

For the source network, NN_S, the choice of architecture and hyperparameters is dependent on the nature of the data and is critical to the performance of the neural network. Model hyperparameters are often selected using a trial-and-error process to maximize performance. In the case of regression, the performance metric for the neural networks is the mean squared error (MSE). To generalize this process, the selection of optimal hyperparameter combinations is automated using Keras Tuner [21] tool from the Keras library. Random search is performed over the hyperparameter space that is defined as:

- Number of hidden layers: $[2,7] \in \mathbb{Z}^+$
- Neurons: $2n: n \in \mathbb{Z}^+, 1 \leq n \leq 25$
- Learning rate: $[10^{-4}, 10^{-2}] \in \mathbb{R}$

The range of values for the hyperparameters is selected to maintain an optimum balance between the accuracy and complexity of the network. Rectified linear activation function or ReLU is selected as the activation function for the hidden layers due to its computational simplicity which makes optimization easier [19]. Adam optimizer [20] is used for its ability to handle sparse gradients on noisy problems. The tuner is evaluated for 10 combinations of hyperparameters over 1000 epochs with 10% data for validation. The model with the lowest validation loss is adjudged as the best model, provided no overfit (substantially higher validation loss than training loss) is observed.

Implementing Transfer Learning for Target Material

Unlike traditional machine learning algorithms, the performance of neural networks improves with the size of training data. Since the data available is limited, training a neural network from scratch for IN718 can be problematic. Transfer learning is especially handy in such data-scarce domains by transferring information from a related domain. The L-DED deposition of source material SS316L and target material IN718 differ only in their thermophysical properties while being governed by the same fundamental physics. Therefore, features learned by a network for SS316L are hypothesized to be transferable to learn the "process parameter-melt pool property" relationship of IN718.

To explore the extent of transferring layers, TL is implemented by sequentially transferring and freezing the weights of the pre-trained layers. This ranges from networks that involve complete retraining of the source network to reusing the pre-trained network with only the output layer being fine-tuned to the target data. Furthermore, the network skill on different sizes of IN718 training-test data is examined by implementing k-fold cross-validation. k-fold cross-validation ensures that

the model performance reported is generalized and not a coincidence of a particular training-test split. The data is split into k subsets of equal size where the network is trained on k-1 subsets. In this paper, k =2, 3, 5, and 10 have been used to evaluate the network performance. The target networks are denoted as $NN_{T[L]_k}$, where L represents the layers at which TL is implemented by transferring and freezing the weights. k represents the number of folds in the k-fold cross-validation. The following metrics are used to evaluate the performance of each network:

i. R^2 : The coefficient of determination quantifies the variation in the dependent variable explained by the independent variable.

ii. MSE: For a predicted melt pool depth, $\hat{d}_i$, and the true, simulated melt pool depth, d_i, at the i^{th} combination of a process parameter, (P_i, v_i), the Mean Squared Error is given by,

$$MSE = \frac{1}{N}\sum_{i=1}^{i=N}\left(\hat{d}_i - d_i\right)^2$$

Here, N is the total number of input-output points.

iii. ε: The relative L^2 norm error is calculated as,

$$\varepsilon = \frac{\sqrt{\sum_{i=1}^{i=N}\left(\hat{d}_i - d_i\right)^2}}{\sqrt{\sum_{i=1}^{i=N}(d_i)^2}}$$

Here, $\hat{d}_i$ is the predicted melt pool depth and, d_i, is the true, simulated melt pool depth, at the i^{th} combination of process parameters, (P_i, v_i).

Results

Source network NN_S using Keras Tuner

A summary of the top 5 neural networks from the 10 networks that are randomly selected by the Keras Tuner is shown in Table 2.

Table 2. A summary of the top five neural networks for SS316L selected by the Keras Tuner.

No. of hidden layers	Learning rate	Neurons							Training loss	Val. loss
		1	2	3	4	5	6	7		
4	0.00019	26	38	12	40	0	0	0	0.0027	0.0030
4	0.00055	22	46	28	38	0	0	0	0.0028	0.0029
4	0.00570	30	10	48	48	0	0	0	0.0029	0.0031
5	0.00233	22	6	6	38	46	0	0	0.0040	0.0041
5	0.00030	46	42	26	4	46	0	0	0.0041	0.0042

Each network is trained and tested on 70% and 30% of the data, respectively. 10% of the training data is set aside for validation. The tuner took 9 mins and 4 seconds to run 10 model evaluations on a 128 GB RAM, Intel® Xeon® Gold 6230 processor, and an NVIDIA Quad Pro P620 GPU. The final network consists of 4 hidden layers. L2 regularization is applied to reduce overfit and ensure smaller generalization errors. The final network architecture and its corresponding learning curves are shown in Figure 3 (a) and (b). Figure 3 (c) shows the performance of the architecture on the test data. Low ε value and high R^2 value confirm the effectiveness of the source network. The predictive errors calculated as the difference between the predicted and true melt pool depths are also analyzed in Figure 3 (d) and are small and randomly scattered around zero.

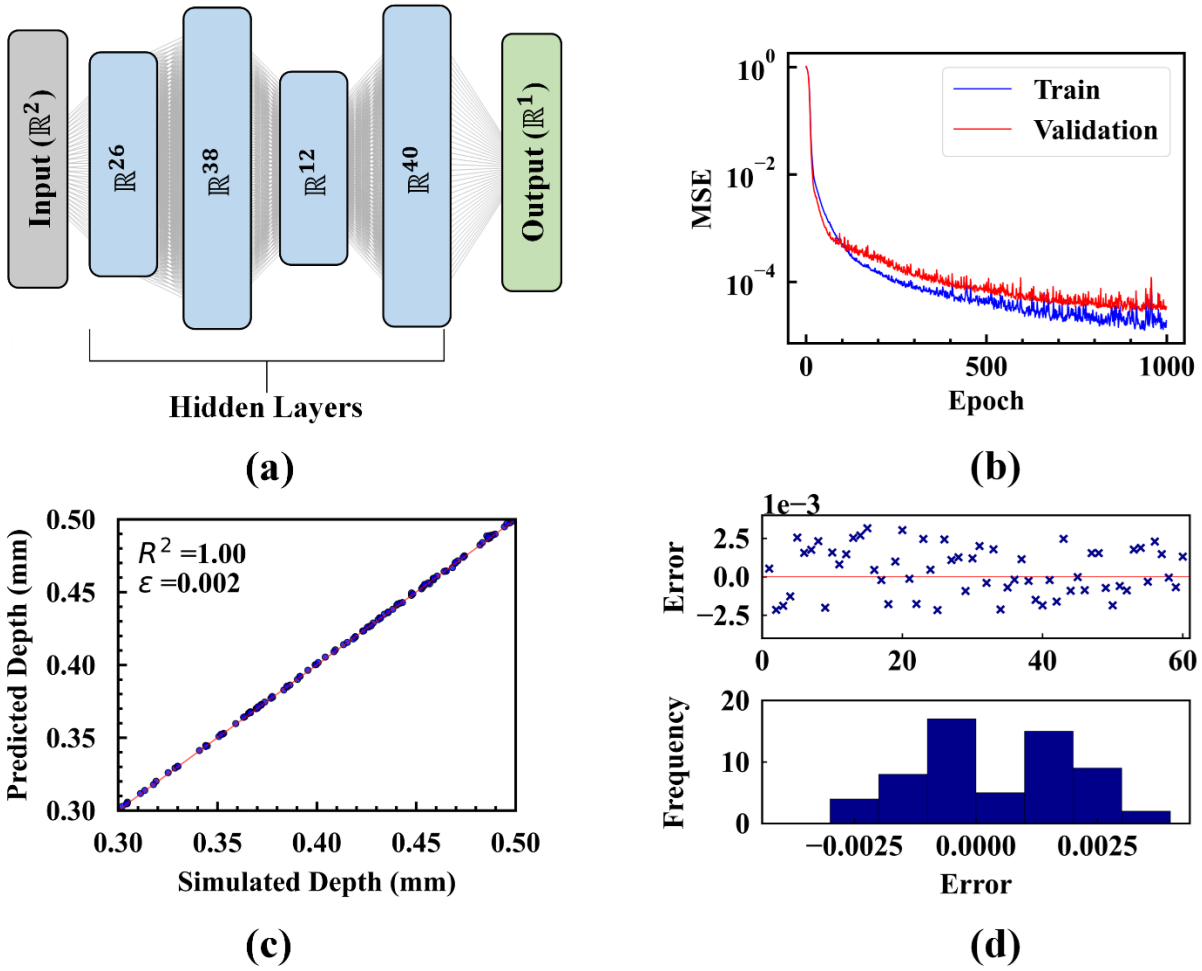

Figure 3. (a) Architecture of the SS316L neural network, NN_S. (b) Learning curves for training NN_S. (c) Parity plot indicating the true depths for the test DoE on the x-axis and the predicted depths on the y-axis. (d) Distribution of the predictive errors.

Transfer Learned Network $NN_{T[L]}$

Once the source network NN_s is developed on the SS316L data, TL is applied for predicting melt pool depth for IN718. The number of layers from the pre-trained NN_s network to be transferred and re-reused on the target IN718 data is determined by sequentially transferring layers. To determine the optimum training-test ratio and evaluate the performance, k-fold cross-validation is

implemented. Each network is trained over 1000 epochs with a batch size of 5. 20% of the training data is kept for validation. Figure 4 shows the results obtained from k-fold cross-validation represented in the form of box plots.

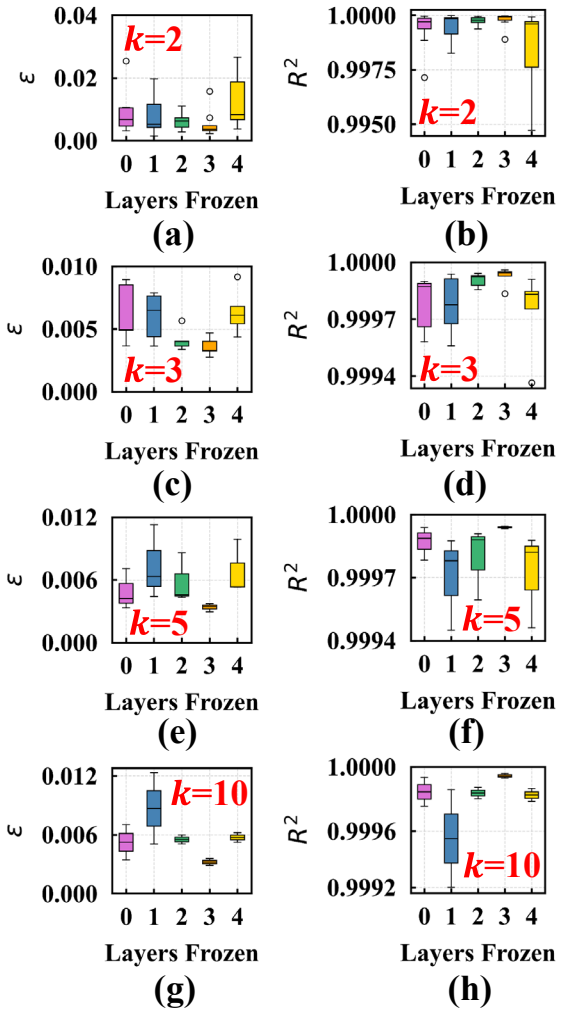

Figure 4. Box plots indicating the results from the k-fold cross validation. The ε, R^2 values of for $NN_{T[L=0,1,2,3,4]_k}$ on the test data are shown in (a) and (e) for k=2, (b) and (f) for k=3, (c) and (g) for k=5, and (d) and (h) for k=10.

Both ε and R^2 are plotted for each network's performance on the test data. Here, '0' refers to a network with no layers repurposed and 4 refers to all but the output layer being repurposed. k=5, which corresponds to a training-test ratio of N_{test} = 20% is observed to work best for the melt pool depth prediction problem in this paper as this corresponds to the lowest MSE and highest R^2 values throughout all cases, without any outliers across the test groups. Additionally, the spread/ variation in the metric values, determined by the length of the box, is also the least for k=5.

A drop in the performance is observed for $NN_{T[4]_{k=2,3,5}}$ evinced by the longer boxes which corresponds to a higher spread in the ε and R^2 values. The R^2 for $NN_{T[4]_{k=2}}$ drops to ~0.86. This reveals that simply fine-tuning the output layer trained on the source SS316L data is too specific

to be retrained accurately on limited IN718 training data. The performance, however, recovers for $NN_{T[4]_{k=10}}$ due to an increase in the training data. Based on the results in Figure 4, a transfer learned network with 3 layers repurposed and evaluated on $k = 5$, i.e., 80-20% training-test data is an appropriate choice for the data considered in this paper. This yields a mean ε of 0.003 and R^2 of 0.999.

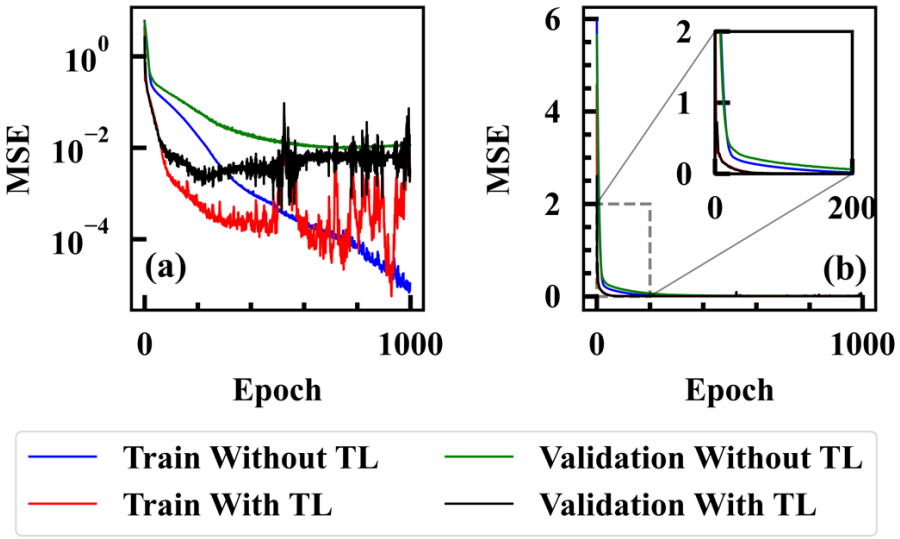

Figure 5. Comparison of training and validation curves for networks with and without TL plotted (a) with log-scale and (b) as-is.

The learning curves of $NN_{T[0]_{k=5}}$ and $NN_{T[3]_{k=5}}$ further highlight the benefits of using transfer learning. Figure 5 shows the training and validation curves for the networks with and without transfer learning on the target IN718 data. For the sake of comparison, the curves are displayed on a log scale in Figure 5 (a) and as-is in Figure 5 (b). The learning curves converge faster with transfer learning than without. Although noisy, the training and validation curves with transfer learning reach a better convergence within a shorter duration, here 1000 epochs. Additionally, the generalization gap or the gap between the training and validation curve keeps growing with epochs for $NN_{T[0]_{k=5}}$ which is a measure of overfit in the model. Both these observations lead to the conclusion that with limited resources, transfer learning achieves better model efficiency in terms of accuracy and budget.

Discussion

The transfer learning methodology is formulated with the initial assumption that the L-DED for two different materials with dissimilar thermophysical properties would still have transferrable features. Such features may be learned from the same fundamental governing equations that define the thermal history of both materials. Although the results qualitatively corroborate the transferability, a more rigorous method is required to quantitatively explain how transferable the features are. Typically, features of the initial layers of a network are more specific to its dataset while those of later layers are general. Yosinki et al. [22] devised a method to determine the transferability of networks by considering five different networks:

1. $NN_{T[0]}$: Base target network that has the same architecture as source network NN_S but is completely trained on the target data.
2. $NN_{T[L]}$: A transfer learned network where 'L' layers are reused from source network NN_S. The weights of these layers are frozen while the rest of the network is trained on the target data.
3. $NN_{T[L]T}$: A transfer learned network where 'L' layers are reused from the base target network NN_{T0}. The weights of these layers are frozen while the rest of the network is trained on the target data.
4. $N^{+}_{T[L]}$: Like $NN_{T[L]}$ but instead of freezing the weights of 'L' layers, they are fine-tuned to the target dataset.
5. $N^{+}_{T[L]T}$: Like $NN_{T[L]T}$ but instead of freezing the weights of 'L' layers, they are fine-tuned to the target dataset.

The motivation behind investigating these different networks is to monitor the adverse effects of transfer learning that arise due to specialization in the layers of the source network, and co-adaptations between neurons (i.e., neurons are correlated). Factors that affect the transferability of a network are (i) disparity between source and target task; (ii) co-adaptation of neurons in the source network; (iii) specificity in the higher layers of the source network. Figure 6 shows the ε for the five different networks. Naturally, for $L = 0$ of all networks, the ε value is the same as that obtained using $NN_{T[0]}$. The base target network or $NN_{T[0]}$ yields an ε of 0.209. $NN_{T[L]T}$ is similar to the base target network as expected with a marginal reduction in ε that resulted from the decrease in network complexity by freezing initial layers. By allowing transferred weights to be fine-tuned, $N^{+}_{T[L]T}$ improves the accuracy of predictions. The increase in ε at layer 4 may be due to stronger co-adaptations between the third and fourth layers that could not be improved by fine-tuning. $N^{+}_{T[L]}$, which is the network with weights transferred from the source network, displays a more generalizable performance with similar trends in ε across the layers. Loosely learned features can fine-tune themselves to overcome the implications of co-adaptations. It can be said that $N^{+}_{T[L]T}$ and $N^{+}_{T[L]}$, by allowing for errors to be backpropagated and the layers to relearn can result in improved performance with lower computational consumption. Finally, the transfer learned to network with layers frozen, $N_{T[L]}$, shows the least ε until at layer 4, at which the source network is too specialized for the source data set. While transferring features boosts the generalization of performance, for the present source and target data set of SS316L and IN718, the use of $NN_{T[L]}$ is, therefore, justified because:

(i) Co-adaptations for the target network are less invasive
(ii) Since target data is scarce, freezing the network layers works better than fine-tuning the network as the latter may lead to overfitting.
(iii) The layers of the source network, except for the final layer, are more generic than specific to the source data.

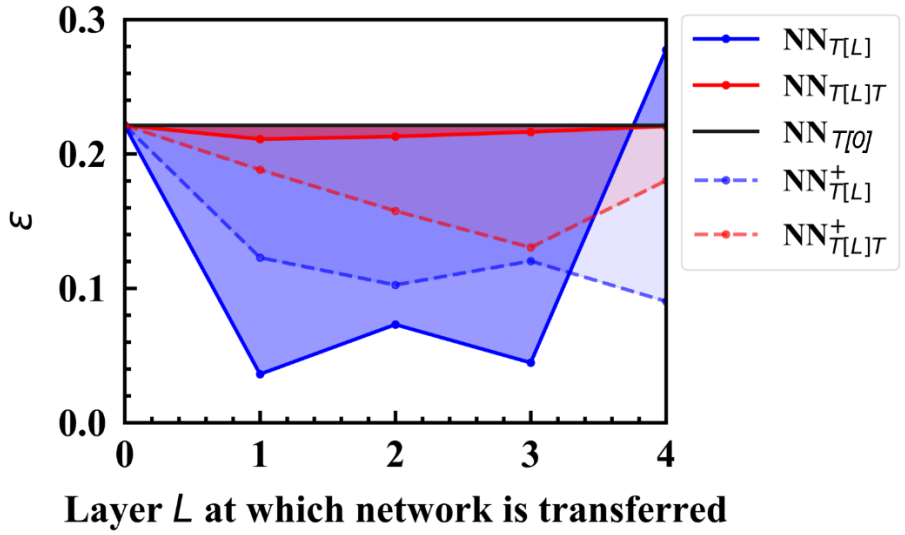

Figure 6. Lines connecting the ε of each treatment for every layer at which the network is transferred.

Conclusions

In this paper, the isolated learning problem associated with process mapping performed via the traditional design of experiments and machine learning techniques is addressed. A transfer learning methodology is demonstrated, using regression models learned for mapping process parameters to melt pool depth for one alloy, here SS316L is transferred to develop the process maps for another alloy, here IN718. The transfer learning framework is proven to be an effective tool in training a large network when the target dataset is significantly smaller (10% of the source dataset) than the source dataset without overfitting. For a train-test ratio of 80-20%, the transfer learned network, with three layers repurposed, has a better generalizable accuracy indicated by a mean $\varepsilon =$ 0.003 and R^2 ~1.0 on the predicted test data predicted. Similarly, the generalization gap in the learning curves is absent for the transfer learned network indicating the absence of any overfit. A neural network without any transfer learning, still had considerable overfitting while being trained on the same number of epochs as the transfer learned network.

The results show the potential of the transfer learning approach in transferring knowledge from process maps for one alloy to generate another. The transferability of the network layers is also investigated to further justify the use of transfer learning on the data. Although this paper considers a relatively inexpensive Eagar Tsai model and, hence, the computational cost is not of a concern, TL can significantly lower the cost of simulation when more complex, expensive computational models are involved. A future study will implement results from experiments into the transfer learning framework. The study can also be extended to transfer process mapping knowledge across different DED systems, e.g., powder-fed L-DED and wire-fed L-DED.

Acknowledgments

The work reported in this paper is supported in part by the Department of Mechanical Engineering at the Pennsylvania State University, University Park, PA 16802. Nandana Menon would like to

acknowledge the James E. Marley Fellowship, the Kulakowski Travel Grant awarded by the Mechanical Engineering Department of the Pennsylvania State University and the National Science Foundation Student Support offered by the organizing committee of the Solid Freeform Fabrication Symposium. Any opinions, findings, and conclusions in this paper are those of the authors and do not necessarily reflect the views of the supporting institutions.

References:

[1] B. Bax, R. Rajput, R. Kellet, and M. Reisacher, "Systematic evaluation of process parameter maps for laser cladding and directed energy deposition," *Addit. Manuf.*, vol. 21, pp. 487–494, 2018.

[2] J. Gockel and J. Beuth, "Understanding Ti-6Al-4V microstructure control in additive manufacturing via process maps," in *24th International SFF Symposium - An Additive Manufacturing Conference, SFF 2013*, 2013.

[3] J. L. Beuth, N. W. Klingbeil, and J. D. Gockel, "Process mapping of cooling rates and thermal gradients." Google Patents, 10-Apr-2018.

[4] J. Gockel, J. Fox, J. Beuth, and R. Hafley, "Integrated melt pool and microstructure control for Ti–6Al–4V thin wall additive manufacturing," *Mater. Sci. Technol.*, vol. 31, no. 8, pp. 912–916, 2015.

[5] L. Johnson *et al.*, "Assessing printability maps in additive manufacturing of metal alloys," *Acta Mater.*, vol. 176, pp. 199–210, 2019.

[6] P. Aggarangsi, J. L. Beuth, and D. D. Gill, "Transient changes in melt pool size in laser additive manufacturing processes," in *2004 International Solid Freeform Fabrication Symposium*, 2004.

[7] A. Vasinonta, J. L. Beuth, and R. Ong, "Melt pool size control in thin-walled and bulky parts via process maps," in *2001 International Solid Freeform Fabrication Symposium*, 2001.

[8] Z. Zhang, Z. Liu, and D. Wu, "Prediction of melt pool temperature in directed energy deposition using machine learning," *Addit. Manuf.*, vol. 37, p. 101692, 2021.

[9] F. Caiazzo and A. Caggiano, "Laser Direct Metal Deposition of 2024 Al Alloy: Trace Geometry Prediction via Machine Learning," *Materials* , vol. 11, no. 3. 2018.

[10] S. Mondal, D. Gwynn, A. Ray, and A. Basak, "Investigation of Melt Pool Geometry Control in Additive Manufacturing Using Hybrid Modeling," *Metals* , vol. 10, no. 5. 2020.

[11] N. Menon, S. Mondal, and A. Basak, "Multi-Fidelity Surrogate-Based Process Mapping with Uncertainty Quantification in Laser Directed Energy Deposition," *Materials (Basel).*, vol. 15, no. 8, p. 2902, Apr. 2022.

[12] X. Huang, T. Xie, Z. Wang, L. Chen, Q. Zhou, and Z. Hu, "A Transfer Learning-Based Multi-Fidelity Point-Cloud Neural Network Approach for Melt Pool Modeling in Additive Manufacturing," *ASCE-ASME J Risk Uncert Engrg Sys Part B Mech Engrg*, vol. 8, no. 1, Aug. 2021.

[13] S. J. Pan and Q. Yang, “A survey on transfer learning,” *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, 2009.

[14] T. W. Eagar and N. S. Tsai, “Temperature fields produced by traveling distributed heat sources,” *Weld. J.*, vol. 62, no. 12, pp. 346–355, 1983.

[15] B. Zheng, Y. Zhou, J. E. Smugeresky, J. M. Schoenung, and E. J. Lavernia, “Thermal Behavior and Microstructure Evolution during Laser Deposition with Laser-Engineered Net Shaping: Part II. Experimental Investigation and Discussion,” *Metall. Mater. Trans. A*, vol. 39, no. 9, pp. 2237–2245, 2008.

[16] Y. Lee, M. Nordin, S. Babu, and D. Farson, “Influence of Fluid Convection on Weld Pool Formation in Laser Cladding,” *Weld. J.*, vol. 93, pp. 292s-300s, Aug. 2014.

[17] J. Hertz, A. Krogh, and R. G. Palmer, “Introduction to the Theory of Neural Computation,” *Introd. to Theory Neural Comput.*, pp. 1–327, Jan. 2018.

[18] F. Chollet, “Keras,” *https://keras.io*, 2015. .

[19] X. Glorot, A. Bordes, and Y. Bengio, “Deep sparse rectifier neural networks,” in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 315–323.

[20] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv Prepr. arXiv1412.6980*, 2014.

[21] T. O’Malley, E. Bursztein, J. Long, F. Chollet, H. Jin, and L. Invernizzi, “KerasTuner.” [Online]. Available: https://github.com/keras-team/keras-tuner. [Accessed: 19-Jun-2022].

[22] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, “How transferable are features in deep neural networks?,” *arXiv Prepr. arXiv1411.1792*, 2014.