Sensors and Materials, Vol. 36, No. 7 (2024) 2835–2849
MYU Tokyo

S & M 3704

Virtual Foundry Graphnet for Predicting Metal Sintering Deformation

Rachel (Lei) Chen,[1*†] Chuang Gan,[2†] Juheon Lee,[1]
Zijiang Yang,[2] Mohammad Amin Nabian,[3] and Jun Zeng[1]

[1]HP Inc., 1501 Page Mill Rd, Palo Alto, CA 94304, USA
[2]HP Inc., Block B, 2727 JinKe Rd, ZhangJiang High-tech Park, Shanghai 201203, China
[3]Nvidia Inc, 2788 San Tomas Expy, Santa Clara, CA 95051, USA

(Received January 3, 2024; accepted July 11, 2024)

Keywords: graph neural networks, Physics-ML, additive manufacturing, digital twin, metal sintering, deep learning

Metal sintering is a necessary step for the fabrication of metal-injection-molded parts and binder jetting techniques such as those in HP's metal 3D printer (MetJet). This process induces significant deformations, typically ranging from 25 to 50%, depending on the porosity of the green part (printed part before post-processing). Achieving precise geometrical accuracy and consistency in the final part presents a substantial challenge to increasing the manufacturing yield. This challenge is primarily attributed to the high porosity of green parts produced by MetJet (compared with alternative technologies such as metal injection molding), which can lead to approximately 50% volumetric shrinkage after sintering. Moreover, this shrinkage is nonisotropic as it depends on nonuniform stress built up during sintering, resulting in deformations such as gravitational sag, gravitational slump, and surface drag. In this study, we employ a graph-based deep learning approach to predict the deformation of the part where deformation simulation can be substantially sped up at the voxel level. By utilizing a well-trained metal sintering inferencing engine, the final sintering deformation value can be obtained in a matter of seconds. The tested accuracy on a sample complex geometry achieves a mean deviation of 0.7 um for a 63 mm test part in a single sintering step (equivalent to 8.3 min of physical sintering time) and a mean deviation of 0.3 mm for the complete sintering cycle (~4 h of physical sintering time).

1. Introduction

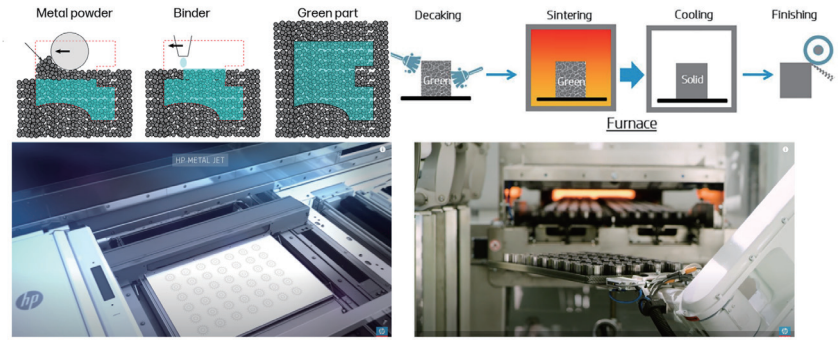
As additive manufacturing advances and expands its applications, the industry stands witness to a diverse portfolio encompassing polymers and metals. As a leading entity, HP, offers a diverse array of products. One standout example is the Metal Jet 3D printing system S100, which employs precision heat control mechanisms to orchestrate the phase transition of metal powder material. This process enables the production of metal components with specified geometries

*Corresponding author: e-mail: lei.chen1@hp.comp
†Contributed equally to this work.
https://doi.org/10.18494/SAM4883

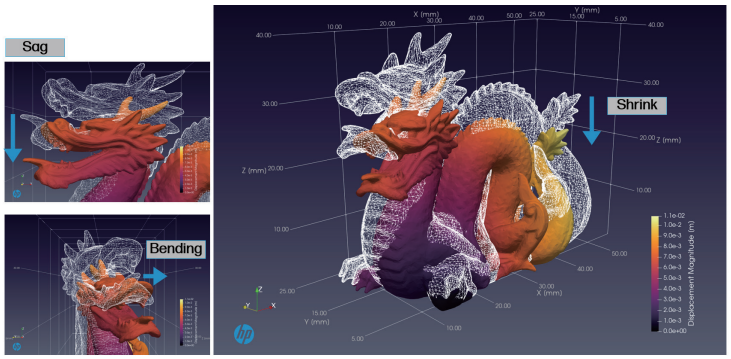
ISSN 0914-4935
https://myukk.org/

and mechanical properties. In tandem, our team is actively engaged in the advancement of digital twin software for simulating the intricate material phase transitions inherent in production processes. This software empowers users with the capability to anticipate and optimize design and process control parameters, thereby facilitating enhancements in product quality and manufacturing yield.

Metal sintering serves as an essential process step in the fabrication of metal-injection-molded (MIM) parts and binder jetting techniques, exemplified by MetJet.[1] As depicted in Fig. 1(a), the MetJet production process involves multiple stages. During metal sintering, significant deformations occur, typically ranging from 25 to 50%, contingent upon the initial porosity of the green part (the component printed and before undergoing post-processing such as sintering to attain the desired mechanical properties).[2,3] The many factors causing the deformation of a green part (e.g., viscoplasticity, sintering pressure, yield surface parameters, yield stress, and gravitational sag) must be identified and used for shape deformation prediction. Achieving the desired geometrical accuracy and consistency in the final part remains a formidable challenge, primarily attributed to two factors. Firstly, the green parts produced by MetJet exhibit a higher porosity than those prepared using alternative technologies such as MIM. Our green parts after sintering exhibited ~50% volumetric shrinkage. Secondly, such shrinkage is not isotropic and depends on the nonuniform stress built up during sintering, e.g., gravitational sag (feature



(a)



(b)

Fig. 1. (Color online) (a) HP's Metal Jet printing stages to obtain the final part. (b) The Stanford dragon test model shows an isotropic shrinkage of the part, including gravitational sag, shrink, and bending.

location disruption due to insufficient support), gravitational slump (downward flow of the material that changes the thickness of the part), and surface drag (manifested as distortion due to friction between the part and the supporting surface during shrinkage-induced movement), as illustrated in Fig. 1(b) for the Stanford dragon test sample.[4]

Existing commercial software for predicting metal sintering deformation includes Abaqus and HP's proprietary software, Virtual Foundry. Virtual Foundry employs domain science-driven predictions based on fundamental principles of sintering physics. It encompasses various factors including thermal profile, yield curve, part porosity, and surface friction to simulate deformation phenomena such as shrinkage and sagging effects. As a physics-based simulation engine, Virtual Foundry currently requires the use of very small sintering time steps in the simulation to manage the nonlinearity inherent in sintering physics. Consequently, simulating the process of transient, dynamic sintering of a part can take from tens of minutes to several hours depending on the part size. Enhancing the speed of first-principles-based simulation tools such as Virtual Foundry could unlock numerous additional applications in metal printing.

In this paper, as a component of our team's digital twin initiative, we propose the Virtual Foundry Graphnet for predicting part deformation. This innovation significantly expedites deformation simulations at the voxel level. With a well-trained metal sintering inferencing engine, only mere seconds is needed to obtain the final sintering deformation value. The tested accuracy on a sample part of complex geometry measuring 63 mm diagonally shows a mean deviation of 0.7 um for a single sintering step (equivalent to 8.3 min of physical sintering time) and a mean deviation of 0.3 mm for the entire sintering cycle (~4 h of physical sintering time). This technology represents our progress toward establishing a self-contained deep-learning framework capable of providing fast (near real time) and accurate end-to-end predictions of sintering deformation of a part. For future work, we are currently investigating the utilization of the real-time printer images obtained from *in situ* thermal or stereovision cameras. By integrating this real-time sensing data into the model, it will have the capacity to accurately reflect predictions specific to variations between printers and even different print chamber locations.

The structure of this paper is as follows. Section 2 provides an overview of recent advancements in digital twin systems, sintering simulation, and research on physics-informed deep learning models. In Sect. 3, we introduce a graph neural network (GNN)-based simulator capable of significantly accelerating sintering simulations, along with implementation details. Testing results on various complex geometries are presented in Sect. 4. We conclude with a summary of the research and propose future directions of optimization, as well as emphasize the open-sourcing of our work to the community while encouraging potential future endeavors.

2. Related Works

Digital twin, as an integrated multiphysics, multiscale simulation, is a virtual replica of an actual physical asset or system. It is a computational model based on physical theories and collected online data and information. It produces accurate and synchronized simulations or forecasts of the future of the physical counterpart.[5–9] The Digital twin system enables

engagement with the physical environment, the updating of changes (performance and maintenance) of the physical system, real-time recommendations and autonomous decisions to facilitate the operation in the physical world through the rapid (real-time) collection of online data, and information analytics. Digital twin extends the Industrial 4.0 revolution into industries such as manufacturing, healthcare, and smart cities.

This work focuses on the simulation of the metal printing process as part of HP's Digital twin efforts. The process of manufacturing a particular material from powder consists of the consolidation step, which involves injection molding, extrusion, slip, isostatic pressing, and so on, and the sintering step that results in the final densified product. The sintering process is the powder metallurgy process during the thermal treatment, in which contacting particles are bonded to form objects with the required geometric and mechanical properties. Sintering occurs primarily at the atomistic level, during which the rheological response of the powder body as a homogeneous, porous continuum can be described by the governing constitutive laws.[10] The sintering kinetics of particulate bodies account for all forms of impact from, for example, the particles themselves, the local interactions with each other, macroscopic factors (e.g., externally applied forces and kinematic constrictions, i.e., the adhesion of the part to the furnace), and the inhomogeneity of the initial density distribution before sintering.

Sintering simulation software tools such as the finite element code Abaqus use the constitutive laws of sintering based on the continuum model. HP's in-house software Virtual Foundry (HP 3D Digital Sintering)[11] also considers the bodies as homogeneous, porous continuums and simulates plastic deformation at elevated temperatures governed by diffusional theories and geometrical models of grains, grain boundaries, and pores. As a physics-based simulation engine, Virtual Foundry currently requires very small sintering time steps in sintering physics simulation, whereby minutes to several hours is required for simulating a part's transient, dynamic sintering process depending on the size of the part, limiting the user applications.

Recent advancements in deep neural networks, as evidenced by their successful application in various domains such as computer vision and natural language processing,[12] have spurred investigations into their utility in scientific problem-solving. For instance, Alphafold[13] has demonstrated successful predictions of protein structures, while FourCastNet[14] has been employed for weather forecasting alongside conventional numerical weather simulation methods. However, the expressive power of neural networks is limited by the size of available data, limiting their feasibility in scientific problems where data acquisition is restricted or costly. To address this challenge, physics-informed neural networks (PINNs) have been developed by incorporating prior knowledge of physics into neural network architectures.[15] PINNs augment conventional neural networks with additional constraints, termed physics-informed loss, to solve systems of partial differential equations (PDEs). As an example, DeepONet uses two networks, TrunkNet and BranchNet; BranchNet is used for encoding and processing the data, and TrunkNet is used for physics-informing the model at any query point. The Fourier neural operator transforms data from the physical domain to the frequency domain and then applies convolutions to the Fourier domain, thus building dimension-free data-driven neural physics models. However, PINNs, DeepONet, and the Fourier neural operator are based on the Euclidean

grid structure. Therefore, it is not easily scalable to scientific problems defined on non-Euclidean domains, such as social networks, sensor networks, genetics, and meshed surfaces.[16–18] To address this limitation, the use of GNNs and their variants[19] has been proposed by applying convolutions to the graph domain such that one can solve the PDE system on non-Euclidean domains.

For complex physical simulation tasks, such as weather systems and fluid dynamics, the objects processed are often distributed in non-Euclidean spaces and contain complex physical interactions. GNNs can represent such data on graph space and simulate object interaction better than conventional convolutional or recurrent neural networks. Because of their powerful data representation and inference capabilities, GNNs have been widely applied in physical simulation with graph-structured data. Wu summarized earlier works on learning node representations with neural architectures.[20] Bruna *et al.* and Kipf and Welling proposed spectral convolution network architectures that use the graph Laplacian spectrum to perform convolutions in the spectral domain.[21,22] Atwood and Towsley and Veličković *et al.* presented spatial convolution networks with simple architectures that use message passing to aggregate local information from neighborhoods and demonstrated state-of-the-art results on some graph tasks.[23,24] In our work, we use spatial convolution networks based on these predecessors' works.[25]

3. Materials and Methods

We propose a GNN-based simulator to construct a quantitative model capable of learning the entire physics simulation process and delivering much faster predictions. The deep neural network is capable of the following.

1. The proposed system directly infers the deformation state of the sintered object (i.e., displacement vectors at the voxel level and other physical fields associated with voxels, such as porosity) at time $T(t = k)$ when provided with the state of the object deformation at time $T(t = 0)$ to $T(t = k - 1)$, $T(t = k - 1) < T(t = k)$, $k > 1$ being the input length. In our testing setting, k was set equal to 5.
2. The proposed system can roll out to infer the sintered object deformation state after $T(t = k)$, i.e., $T(t = k + 1)$, $T(t = k + 2)$, … $T(t = k + l)$, $l > 1$ being the prediction length, given the state of the object deformation at $T(t = 0)$ to $T(t = k - 1)$. We observed that in our tested setting, l could be larger than 50 and possibly even more with optimized model training, enabling the coverage of the entire sintering deformation simulation with promising prediction accuracy.
3 The proposed system infers the sintering simulation state at $T(t = k)$ using the sintering simulation state from $T(t = 0)$ to $T(t = k - 1)$ as input and achieves a near real-time performance of, for example, less than 1 s. It predicts all rollout simulation states, i.e., $T(t = k + 1)$, $T(t = k + 2)$, … $T(t = k + l)$, $l > 1$, in less than or around 10 s (depending on the input geometry size) with one NVIDIA GPU. This is much faster than obtaining each sintering state through finite element analysis (FEA) simulation, which requires many time steps, each of which may take minutes to complete. Using our proposed inferencing model to replace physical simulation achieves a much faster speed-up for Virtual Foundry.

3.1 Data preparation

Figure 2 illustrates the data processing flow to convert the voxelated parts data from the physics-based simulation engine to obtain the sintering-time graph data for model training and inferencing. The complete data preparation pipeline is presented in Fig. 3. First, Virtual Foundry transfers the origin design files (e.g., STL, OBJ, and 3MF) of parts to voxel data through voxelization and outputs a voxelated format representing parts. Using the voxelated data, we compute vertices and edges of voxel data to build a graph $\boldsymbol{G}(\boldsymbol{V}, \boldsymbol{E}, \boldsymbol{U})$. $\boldsymbol{G}$ represents a graph structure model at a specific sintering time step. $\boldsymbol{V}$ represents the nodes and $\boldsymbol{E}$ represents the edges between two nodes. The connection of two nodes through the edge represents the interactions between the two nodes.

Each node in the graph has a list of attributes, as shown in Fig. 4. This list indicates the features of this node. For example, the node attribute list of node i is in the form of $[\boldsymbol{s}_{k-2}, \boldsymbol{s}_{k-1}, \boldsymbol{s}_k, T_k]^T$, where $\boldsymbol{s}_{k-2}$, $\boldsymbol{s}_{k-1}$, and $\boldsymbol{s}_k$ are the velocities of the previous three time steps of node i. Each $\boldsymbol{s}_k$ is a three-dimensional vector describing velocity in XYZ-dimensions. Velocity vectors for the initial n time steps ($n > 1$ being the input length, the equation shows the case where $n = 3$; we performed the test with various n values such as $n = 2$ and 6) are added as node attributes representing the moving speed of each node. Sintering physics has an intrinsic memory effect; the current state strongly depends on the history of the previous state.

Other physics-inspired parameters can be added as node attributes, such as node velocity and acceleration vector of each time step. Boundary constraints were also implemented to distinguish nodes that can move freely from those that fall on the contacting surface.

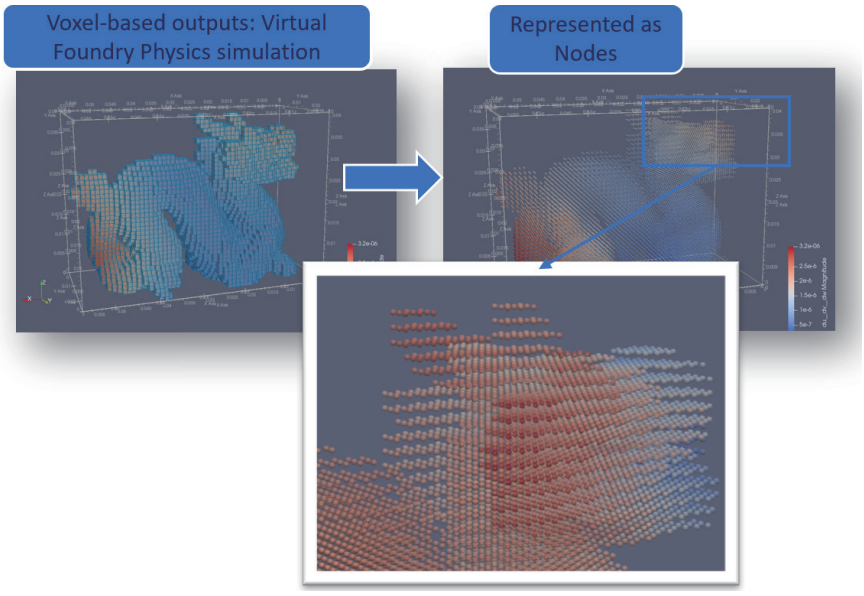


Fig. 2. (Color online) Data processing flow to obtain the sintering-time graph data for training and inferencing. The sample dragon built is voxelated where the color gradient shows each voxel's deformation scale. We preprocess the voxel-based simulation output (left) to form "nodes" of the graph (right), representing the concept of "metal particles".

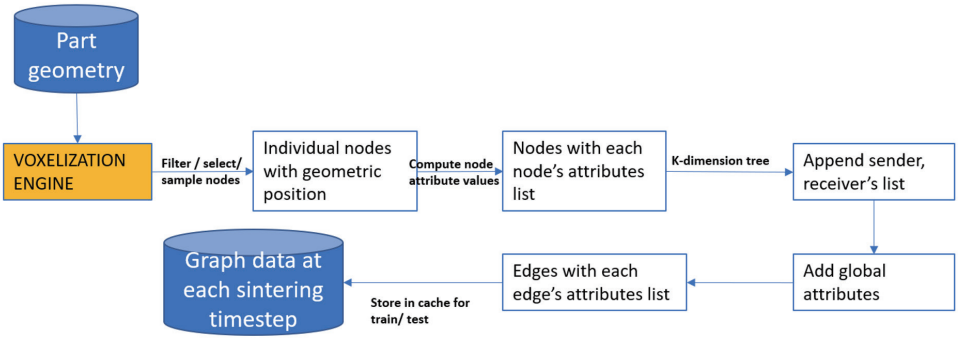


Fig. 3. (Color online) Data processing flow to obtain the sintering-time graph data for training and inferencing. The preprocessing steps are for converting the voxel-based simulation data into a graph-based data structure for each sintering step

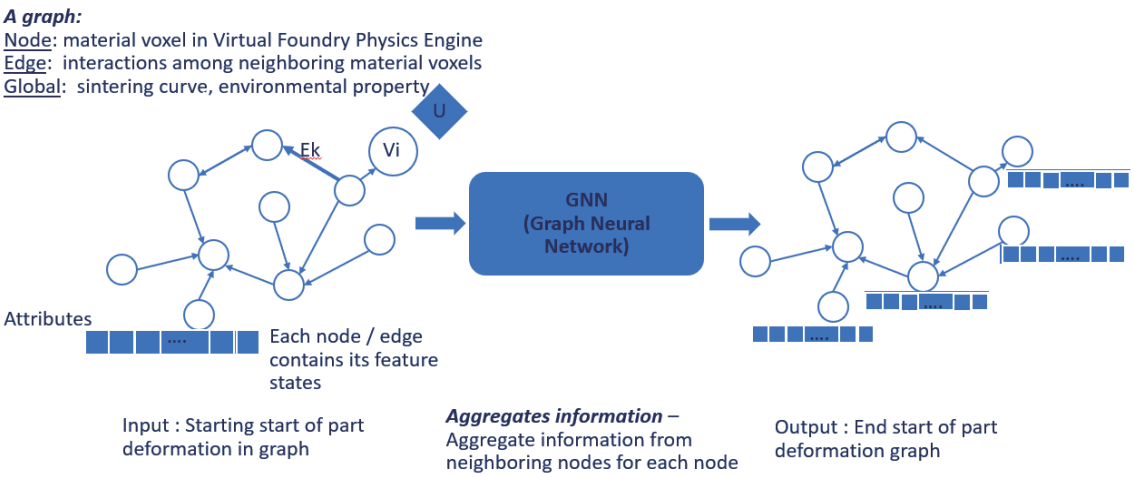


Fig. 4. (Color online) Simulation prediction using the processed graph data with the GNN architecture.

To define the neighbors of each node, we built directed edges among nodes using a k-dimensional tree and found neighbors of each node within radius r. Node $\boldsymbol{V}_i$ was set as the **"sender"** of a directed edge; nodes within the range of its radius r are included as the **"receivers"** of the directed edge of this node. We defined radius r as 1.2 times voxel size, so each node has six neighbors as receivers. Each edge in the graph also has a list of attributes, including the relative distance between the connected nodes.

Optionally, we implemented an additional component, $\boldsymbol{U}$, representing the global factors of the entire graph. For example, we used sintering temperature at different time steps as component $\boldsymbol{U}$.

At different sintering time steps, the features of the graph structure change, such as the number of edge connections among nodes (indicating the dynamics of node interactions) and

attribute values of nodes and edges (indicating the change in displacement value at each node and the change in stress).

3.2 Model architecture and training

The learned simulator consists of three stages: encoder, processor, and decoder. The encoder converts Virtual Foundry data to latent graph data $\boldsymbol{G}(\boldsymbol{V}, \boldsymbol{E}, \boldsymbol{U})$. The processor computes interactions among nodes, aggregates information, and outputs learned latent features. The processor has an interaction networks module. Firstly, it uses the features from the previous step to update edge features (edge attributes list). Then, it uses the previous feature set and the updated edge features to update node features. Finally, it uses previous, edge, and node features to update global features.

We employ a multilayer perceptron (MLP) for each update function. The weights and parameters of each MLP are learned during training. Multiple rounds of this calculation are performed; each round is referred to as one "message passing" step, representing the interaction among nodes through edges. The number of message passing rounds can vary depending on the training set. In our case, we set it to 10. The number of message passing rounds is a hyperparameter that can be tuned considering the trade-off between the model prediction accuracy (since more message passing rounds represent an enormous node interaction scope) and the model runtime. We choose the optimal trade-off setting between the accuracy and the training runtime. The decoder extracts node feature data (in the form of node acceleration or displacement vector) from learned latent features and sends it to Virtual Foundry.

3.3 Objective functions

Our loss function design is as follows. At time point k, features with length n are input to GNNs (f) and node accelerations with length l are output from GNNs f. Instead of predicting one-step acceleration after time point k, we predicted and constrained node accelerations of multiple steps. That is, the graph networks could directly or recurrently output the elements of $\boldsymbol{Y}$:

$$\begin{aligned} \boldsymbol{Y} &= \left[\boldsymbol{a}_{k+1}, \boldsymbol{a}_{k+2}, \boldsymbol{a}_{k+3}, ..., \boldsymbol{a}_{k+l}\right], \\ \boldsymbol{x} &= \left[\boldsymbol{s}_{k-n}, ..., \boldsymbol{s}_{k-2}, \boldsymbol{s}_{k-1}, \boldsymbol{s}_{k}, \boldsymbol{g}\right], \\ \boldsymbol{Y} &= f_{\theta}\left(\boldsymbol{x}\right). \end{aligned} \tag{1}$$

Metal sintering is a complex physics process, and predicting a long-term deformation without a substantial dataset proves challenging. Instead of predicting one step after time point k, our method predicts l consecutive small steps, where l can be one step or multiple steps. Multiple steps are selected to prevent the graph networks from learning the short-term dynamics of sintering. This can potentially alleviate the overfitting of training and reduce the prediction time of rollout. Mean square error is used to optimize the graph networks, and a discount factor is added to the loss, where $\boldsymbol{a}'$ denotes the predicted values and $\boldsymbol{a}$ denotes the ground truth values.

$$\boldsymbol{L}(\boldsymbol{Y} \mid \boldsymbol{x}) = (\boldsymbol{a}'_{k+1} - \boldsymbol{a}_{k+1})^2 + \gamma(\boldsymbol{a}'_{k+2} - \boldsymbol{a}_{k+2})^2 + \gamma^2(\boldsymbol{a}'_{k+3} - \boldsymbol{a}_{k+3})^2 + \ldots + \gamma^{l-1}(\boldsymbol{a}'_{k+l} - \boldsymbol{a}_{k+l})^2 \quad (2)$$

Additionally, aside from differentiating the node types and adding the type of vector as a node attribute for the model to learn differentiation during training, we also constrained the "Fixed node" and "Slip nodes" moving explicitly by setting their velocity or acceleration equal to 0. These were added as additional constraint terms in the model loss function. With this loss function implementation, the model learns explicitly that different node types have 0 acceleration in the stated dimensions, that is, they are not moving in the stated dimensions.

3.4 Inferencing

Similarly to the data preparation in the model training process, the converted sintering state graphs for the initial several time steps serve as the first input dataset to the trained graph networks. Corresponding to the training set, the model takes one data sample from $t = 0$ to $t = n$ (the initial n steps, where n is equal to the model training parameter setting) as input, and $\boldsymbol{G}(\boldsymbol{V}, \boldsymbol{E}, \boldsymbol{U})$ at $t = n + 1$ is to be predicted. Firstly, the trained graph network makes the prediction for the next time step. Then, the output of trained graph network $\boldsymbol{G}(\boldsymbol{V}, \boldsymbol{E}, \boldsymbol{U})$ at $t = n + 1$ is recursively fed as input, which builds up the following time step's graph input (from $t = 1$ to $t = n + 1$) to the same trained graph networks to predict the next time step's ($t = n + 2$) graph features. This process is iterated until the entire sintering process prediction is completed, as illustrated in Fig. 5.

4. Experimental Results

We used NVIDIA GPUs to train and test the model. The model was implemented with Pytorch with parallel training and mixed precision. Our code has been integrated into the NVIDIA modulus platform and serves as an example of the open-source application of the Physics-ML models.

4.1 Model optimization

We trained the proposed model on a limited number of geometries (seven different geometries at a mesh size equal to 1000 μm are included in the training data to produce the testing results

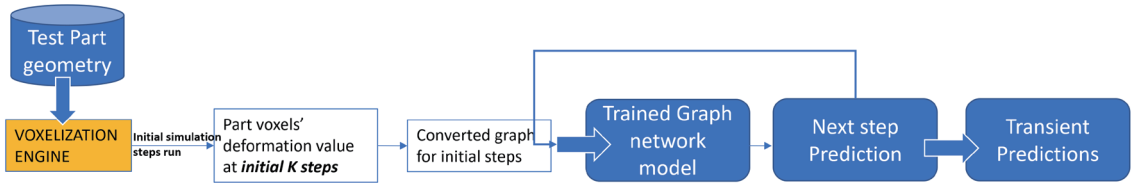


Fig. 5. (Color online) Inference flow to iteratively predict the next sintering state.

given in Table 1). We tested both the "1-step accuracy" (the prediction accuracy of the very next time step) and the "rollout accuracy" (the prediction accuracy after the entire sintering process) on different geometries. Table 1 shows the version optimizations based on our baseline graph networks model. The improved model versions and their improved measured accuracies are listed. We tested model architecture and hyperparameter variations, including adding sintering temperature as a global feature, implementing edge dropout (with the random edge dropout ratio set to 60%, the randomly selected edges are updated for every training epoch) during model training, normalizing the input node features, including node velocities and accelerations, and adding additional anchoring point losses. Model version #4 shows the highest prediction accuracy in both the next step and rollout predictions. We adopted the best model architecture version in later tests and report the test accuracy in the next section.

4.2 Test geometry accuracy

We demonstrate the model testing accuracy on four different test geometries and different voxelization resolutions (voxel size equal to 1000 and 500 μm). The details of the test parts, including the number of voxels and the number of nodes for each test geometry after data preprocessing, are shown in Table 2. Figure 6 shows the simulation completed state result of the part deformation compared with that at the start of sintering (no deformation). The color bar shows the displacement magnitude in millimeters.

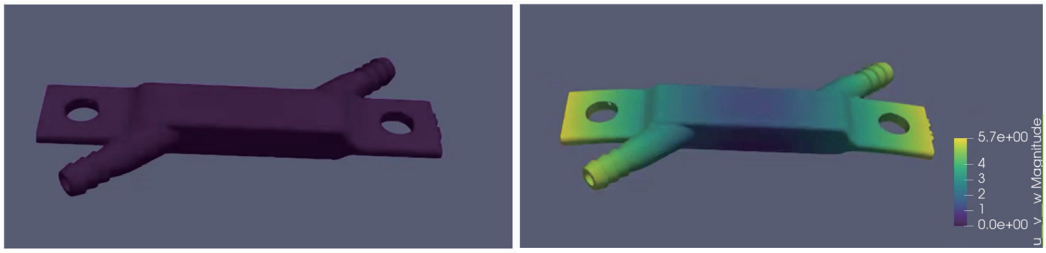
We show the rollout prediction accuracy on two tested parts in Fig. 7. The bar plot shows the mean nodal prediction accuracy (measured as a mean error in millimeters) until the end of the end-of-sintering cycle, and the dot plot in red shows the maximum nodal prediction accuracy (measured as a mean error in millimeters). The mean nodal predicted difference for the first test part (the USB-shape part) is below 0.3 mm, and the maximum nodal predicted difference is

Table 1
Accuracies of model versions.

	Model version/accuracy on: Test part #1/Test part #2	*T*-profile	Edge dropout	Acce and velocity normalization	Anchor loss	1-step MSE ($\times 10^{-9}$)	Rollout MSE ($\times 10^{-2}$)
1	GNS + temperature (*T*)	Yes	—	—	—	49.7 / 95.7	52.1 / 43.8
2	GNS + *T* + edge dropout	Yes	Yes	—	—	5.9 / 1.6	0.3 / 0.1
3	Model 2 + node feature normalization	Yes	Yes	Yes	—	1.9 / 0.2	0.1 / 0.5
4	Model 3 + anchoring loss	Yes	Yes	Yes	Yes	1.5 / 0.6	0.1 / 0.1

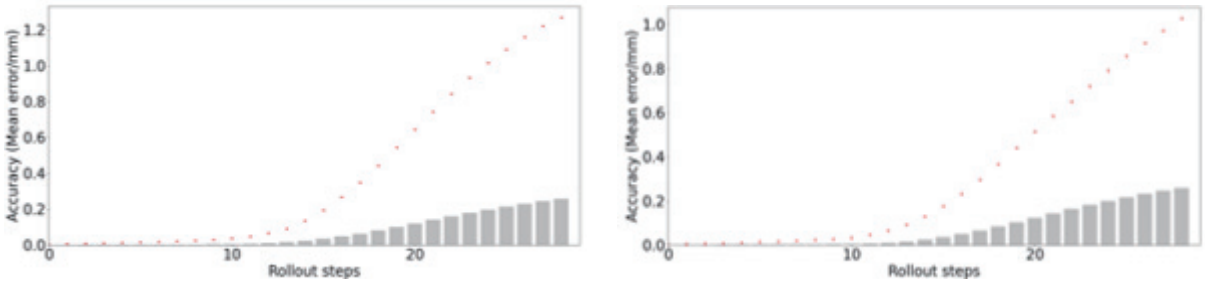
Table 2
Model sizes for different geometries.

Test	Geometry/size (voxels/nodes count)		
	Mesh size = 1000 (10^3)	Mesh size = 500 (10^3)	Size (diagonal/mm)
Extrusion screw	0.8 / 6.8	4.8 / 38.7	19
USB	5.5 / 44.3	35.3 / 282.0	59
Pushing grip	10.6 / 84.4	79.9 / 639.4	87
Busbar	3.8 / 30.6	20.5 / 11.8	63



(a) (b)

Fig. 6. (Color online) (a) The sample geometry of the busbar at sintering start time shows no deformation across the part. (b) Part deformation after sintering. The simulation was generated with a mesh size equal to 500 μm.



(a) (b)

Fig. 7. (Color online) Rollout prediction accuracy on two tested geometries. (a) USB. (b) Busbar. The bar plot shows the mean nodal prediction accuracy, and the dot plot (in red) shows the maximum nodal prediction accuracy.

around 1.2 mm. For the second test part (busbar), the mean nodal predicted difference is around 0.2 mm and the maximum nodal predicted difference is around 1 mm at the end of the sintering simulation process. Figure 8 illustrates the nodal-level deformation sampling at a mesh size equal to 1000 μm at the sintering start time (with no deformation) and the sintering completion state, as well as the simulation ground truth and the model prediction.

We show the test results for all four geometries at a mesh size equal to 1000 μm in Fig. 9: the USB (part size of 59 mm, measured along the diagonal length), busbar (part size of 63 mm), extrusion screw (part size of 19 mm), and pushing grip (part size of 87 mm). The inference model demonstrated good scalability with a minor accuracy tradeoff at smaller mesh sizes (we conducted tests on sample parts with mesh sizes of 500 and 300 μm) in our tests. Figure 9 shows inference runtimes within or around 1 min for obtaining the final deformation prediction for all four parts, computed using the input generation time from Virtual Foundry and the trained network inferencing time. Two tested parts exhibited a maximum node error of less than 2% (relative to the part size). The pushing grip exhibited the most significant maximum nodal error, yet it remained within 5%. This inference run simulated the actual physical sintering time of approximately 3 h.

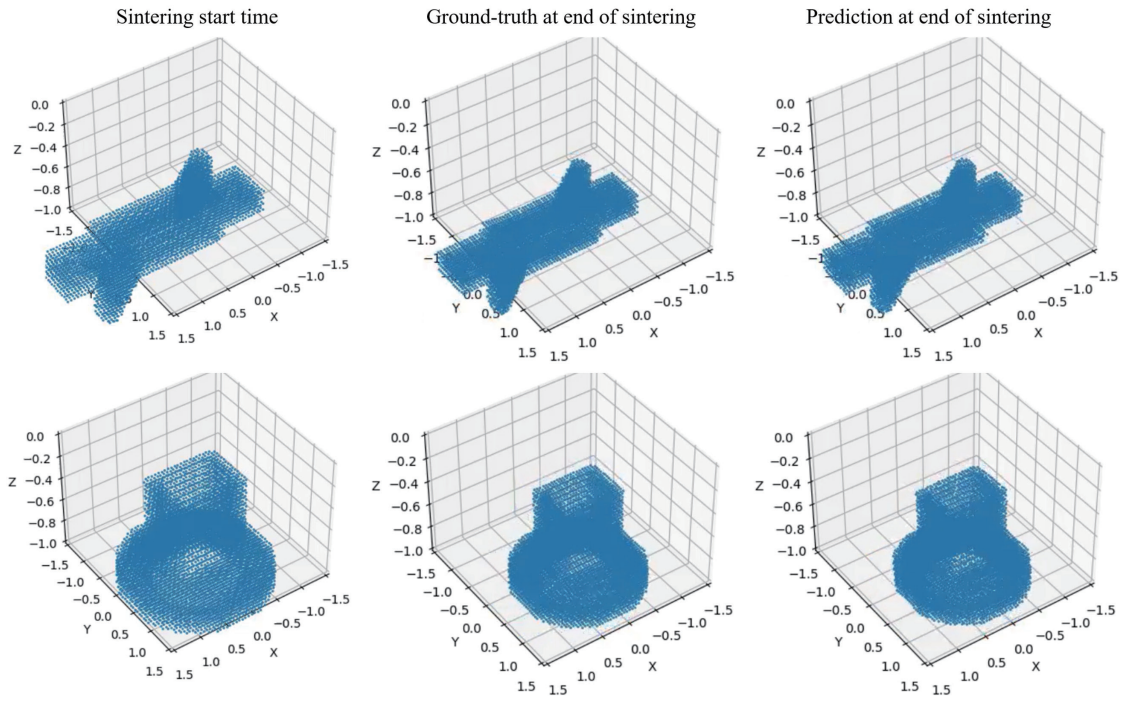


Fig. 8. (Color online) Visual representation using point data for two evaluated geometries. From left to right: initial geometry at the start of sintering, actual geometry at the conclusion of sintering (ground truth), and predicted geometry at the conclusion of sintering.

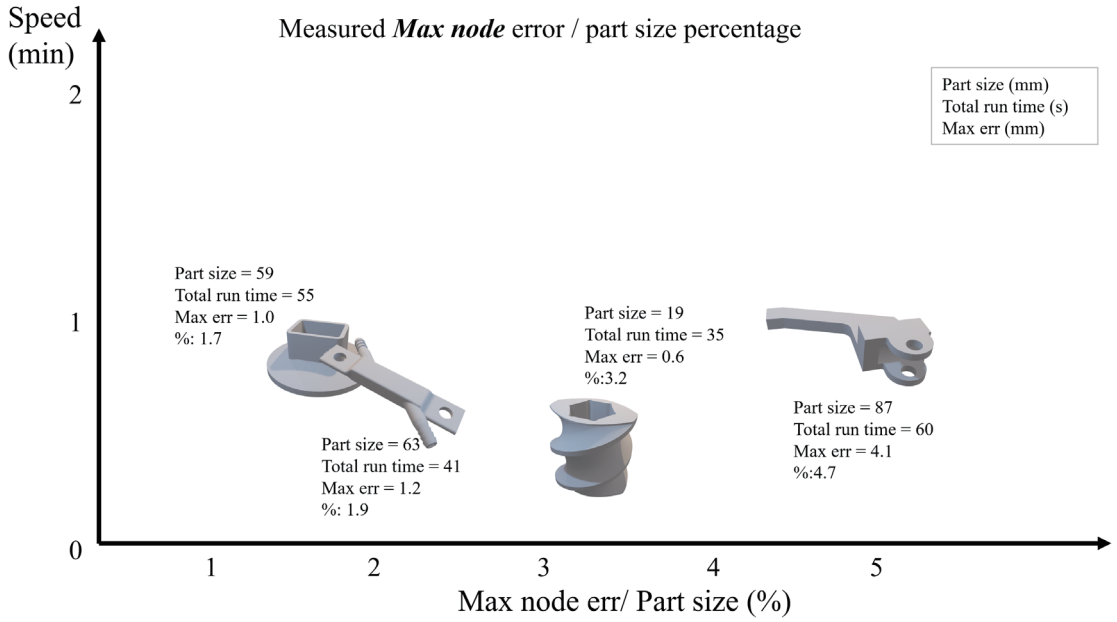


Fig. 9. (Color online) Max node error (in mm and % relative to part size), inference runtime (to obtain the final deformation prediction) vs original part size for the four tested geometries (left to right: USB, busbar, extrusion screw, and pushing grip).

5. Conclusions and Future Work

With a focus on model prediction accuracy together with a high inference speed, in this study, as part of our team's Digital twin initiative, we introduced Virtual Foundry Graphnet. This approach used Physics-ML to notably expedite computations involved in predicting metal powder material phase transitions. It achieved a considerable speed-up compared with conventional physics simulation software while maintaining acceptable accuracy levels. Moreover, Virtual Foundry Graphnet demonstrated promising scalability for parts of varying geometrical complexities and configurations of process parameters. As future tasks, we aim to explore additional approaches to further integrate physics-based insights into the proposed model architecture. For instance, we could augment the model by incorporating more simulated voxel-based part features, such as fine-resolution porosity and surface friction, or by integrating PDEs into the loss function to enhance the model's accuracy in predicting the entire sintering profile.

Furthermore, to bridge the gap between the printed parts and the simulated results, we propose to extract real-time printer images captured by *in situ* thermal or stereovision cameras. By incorporating real-time sensing data into the model, the model will be able to reflect predictions specific to printer-to-printer variations and even print chamber locations.

We recognize the significant role a growing open-source community can play in accelerating the development of Physics-ML PINN and expanding its applications; NVIDIA Modulus is an exemplary open-source framework for constructing, training, and fine-tuning Physics-ML models. By incorporating Virtual Foundry Graphnet into Modulus as an open-source component, we aim to make metal sintering simulation accessible to a wider audience of researchers and engineers. For further details and access to our code, please visit the Modulus website: https://github.com/NVIDIA/modulus.

Acknowledgments

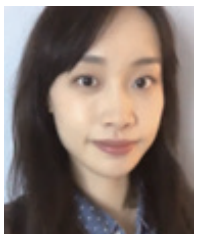
This article is based on the work supported by the Air Force Office of Scientific Research under award number FA9550-23-1-0739. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the United States Air Force.

References

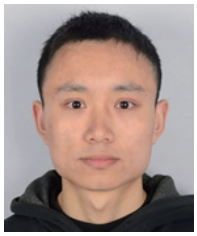
1 HP Inc.: https://h20195.www2.hp.com/v2/GetDocument.aspx?docname=4AA8-1958ENW (accessed June 1, 2024).
2 A. Yegyan Kumar, J. Wang, Y. Bai, S. T. Huxtable, and C. B. Williams: Mater. Design **182** (2019) 108001. https://doi.org/10.1016/j.matdes.2019.108001
3 J. A. Slotwinski and E. J. Garboczi: AIP Conf. Proc. (2014) 1197. https://doi.org/10.1063/1.4864957
4 The Stanford 3D Scanning Repository: https://graphics.stanford.edu/data/3Dscanrep/ (accessed June 1, 2024).
5 A. Fuller, Z. Fan, C. Day, and C. Barlow: IEEE Access **8** (2020) 108952. https://doi.org/10.1109/ACCESS.2020.2998358
6 E. Glaessgen and D. Stargel: 53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conf. (2012). https://doi.org/10.2514/6.2012-1818

7 M. Mashaly: Procedia Comput. Sci. **184** (2021) 299. https://doi.org/10.1016/j.procs.2021.03.039.
8 Z. Liu, N. Meyendorf, and N. Mrad: AIP Conf. Proc. **1949** (2018) 020023. https://doi.org/10.1063/1.5031520
9 A. Madni, C. Madni, and S. Lucero: Systems **7** (2019) 7. https://doi.org/10.3390/systems7010007
10 D. C. Blaine: "The micromechanical influences on the constitutive laws of Sintering for continuum model," Thesis, Pennsylvania State University (2004).
11 HP Inc.: https://h20195.www2.hp.com/v2/GetDocument.aspx?docname=4AA8-3356ENW (accessed July 8, 2024).
12 D. Khurana, A. Koli, K. Khatter, and S. Singh: Multimedia Tools Appli. **82** (2022) 3713. https://doi.org/10.1007/s11042-022-13428-4
13 J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Zidek, A. Potapenko, A. Bridgland, C. Meyer, S. Kohl, A. Ballard, A. Cowie, B. Romera-Paredes, S. Nikolov, R. Jain, J. Adler, T. Back, S. Petersen, D. Reiman, E. Clancy, M. Zielinski, and D. Hassabis: Nature **596** (2021) 583. https://doi.org/10.1038/s41586-021-03819-2
14 T. Kurth, S. Subramanian, P. Harrington, J. Pathak, M. Mardani, D. Hall, A. Miele, K. Kashinath, and A. Anandkumar: Proc. Platform for Advanced Scientific Computing Conf. (2023). https://doi.org/10.1145/3592979.3593412
15 S. Cuomo, V. Cola, F. Giampaolo, G. Rozza, M. Raissi, and F. Piccialli: J. Sci. Comput. **92** (2022) 88. https://doi.org/10.1007/s10915-022-01939-z
16 M. Bronstein, J. Bruna, Y. Lecun, A. Szlam, and P. Vandergheynst: IEEE Signal Process Mag. **34** (2017) 18. https://doi.org/10.1109/MSP.2017.2693418
17 K. Atz, F. Grisoni, and G. Schneider: Nat. Mach. Intell. **3** (2021) 1023. https://doi.org/10.1038/s42256-021-00418-8
18 W. Cao, Z. Yan, Z. He, and Z. He: IEEE Access **8** (2020) 35929. https://doi.org/10.1109/ACCESS.2020.2975067
19 K. Xu, W. Hu, J. Leskovec, and S. Jegelka: Int. Conf. Machine Learning (2019). https://doi.org/10.48550/arXiv.1810.00826
20 Z. Wu: IEEE Trans. Neural Networks and Learning Systems **32** (2021) 4. https://doi.org/10.1109/tnnls.2020.2978386
21 J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun: arXiv preprint. https://doi.org/10.48550/arXiv.1312.6203
22 T. N. Kipf and M. Welling: Int. Conf. Machine Learning (2017). https://doi.org/10.48550/arXiv.1609.02907
23 J. Atwood and D. Towsley: Advances Neural Inf. Process. Syst. 29 (2016). https://doi.org/10.48550/arXiv.1511.02136
24 P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio: Int. Conf. Machine Learning (2018). https://doi.org/10.48550/arXiv.1710.10903
25 A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, and P. Battaglia: Int. Conf. Mach. Learn. **21** (2020) 8459.

About the Authors



Rachel (Lei) Chen is a machine learning research engineer at HP Inc. Her B.S. degree in electrical engineering was obtained from Korea Advanced Institute of Science and Engineering (KAIST) in 2017 and her M.S. degree in electrical and computer engineering from Duke University in 2019. She is now devoted to research on applying cutting-edge deep learning algorithms in additive manufacturing quality control, acceleration, and digital twins. (lei.chen1@hp.com)

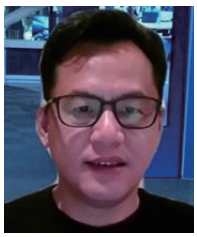


Chuang Gan received his B.S. degree from Nanjing University of Aeronautics and Astronautics, China, in 2014 and his M.S. degree from Zhejiang University, China, in 2017. Since 2017, he has been a software engineer and researcher at HP Inc. His research interests are in deep learning and computer vision. (chuang.gan@hp.com)

Juheon Lee received his PhD degree from University of Cambridge, UK, in 2016. From 2019 to 2023, he worked as a research scientist in HP Inc., during which time his research interests were in geometric deep learning and physics informed neural networks and neural tangent kernel theory. (Juheon.lee.626@gmail.com)

Zijiang Yang received his B.S. degree from Zhejiang University, China, in 1997 and his Ph.D. degree from Zhejiang University, China, in 2003. From 2003 to 2008, he was a senior software specialist at Nokia, China. Since 2008, he has been a software expert and master technologist at HP Inc., China. His research interests are in deep learning and IoT. (zijiang.yang2@hp.com)

Mohammad Amin Nabian is a senior software engineer, AI-HPC at NVIDIA. Mohammad is one of the core developers of NVIDIA Modulus, an AI framework for Physics-ML models. Mohammad received his Ph.D. from the University of Illinois at Urbana-Champaign, with research focused on artificial intelligence for scientific computing. Mohammad's background is in numerical simulation, artificial intelligence, and uncertainty quantification. (mnabian@nvidia.com)



Jun Zeng is a Distinguished Technologist at HP Inc. and a principal investigator and research manager leading software research in 3D printing and digital manufacturing for the HP 3D Printing Software group. His publications include a co-edited book on the computer-aided design of microfluidics and biochips, a co-authored book on the production management of digital printing factories, more than 50 peer-reviewed papers, and 42 granted patents and more pending. His academic training includes advanced degrees in mechanical engineering (PhD) and computer science (M.S.), both from Johns Hopkins University. Jun is an ACM member and an IEEE senior member. (jun.zeng@hp.com)