# A Deep Learning Pipeline for Optimizing Large-scale Phase Field Simulations

Ramakrishnan Kannan[a], Cristina Garcia-Cardona[b], Balasubramaniam Radhakrishnan[c] and Sudip K. Seal[a]
[a]Computer Science and Mathematics Division, Oak Ridge National Laboratory, USA
[b]Computer, Computational and Statistical Sciences Division, Los Alamos National Laboratory, USA
[c]Computational Sciences and Engineering Division, Oak Ridge National Laboratory, USA

***Abstract*—Phase field (PF) simulations are computationally expensive but remain a key analysis tool to understand the complex mechanisms of additive manufacturing (AM) processes. Each PF simulation-aided analysis requires thousands of node hours on leadership-class supercomputers. One of the main goals of these analyses is the study of microstructure evolution during the build process which begins with the onset of nucleation. Nucleation occurs under certain thermo-mechanical conditions which are not known a priori and many PF simulations are required to identify ranges of input thermo-mechanical parameters that can result in the onset of nucleation. Since many of the simulations do not result in nucleation, an analysis campaign often ends up wasting tremendous amounts of precious computing resources executing nucleation-absent simulations. The goal of this work is to design and train deep learning models to inform a PF simulation about the likelihood of the occurrence of nucleation in a future simulation time-step based on the state summary over a finite number of past time-steps of a running simulation. If the prediction determines that the running simulation is unlikely to reach nucleation in the allotted time, then its execution is stopped immediately ultimately resulting in vast reduction in wasted computations when accrued over all the PF simulations typically performed in a single or multiple analysis campaign(s). The paper presents the performance of a machine learning pipeline that uses a convolutional neural network (CNN) model to learn an embedding which is then used with a self-attention network to build a multi-task deep learning model to predict the likelihood of nucleation. The model also predicts the input parameters used in a simulation. Performance is compared with a baseline pipeline that uses an off-the-shelf LeNet-5 model to learn the initial embedding. Despite their smaller size, performance results indicate significant improvement in accuracy of the proposed models compared to the larger baseline models.**

## I. Introduction

The importance of AM for the industry and the government is well-established. In addition to more cost-effective production of components in smaller batches and lowering maintenance costs, other key benefits are energy savings and reduction in both waste and prototyping costs. However, AM is a complex physical process and the path from initiation to a final built-up component goes through a number of sub-processes such as emergence of solidification microstructures. The properties of these microstructure closely determine the mechanical and thermal properties of the final product. For example, in many structural alloys, the solidification microstructure following powder bed-based additive manufacturing consists of columnar cells or dendrites that are elongated along the growth direction (see Figure 1). AM components with such columnar microstructures exhibit anisotropic mechanical properties that are hard to certify for a given application. A case in point is titanium alloys used in aerospace applications. The work horse alloy, in this case Ti-6Al-4V (Ti64), is typically used in most aerospace applications in the wrought form, produced by extensive thermo-mechanical processing that results in more or less spherical or "equiaxed" grain structure in the alloy. Additive Manufacturing (AM) components of Ti64 have to be subjected to expensive post-AM heat treatments to breakdown the columnar structure and recover the isotropy in mechanical properties. However, AM offers significant cost benefit because of the near-net-shape (initial fabrication is close in size and shape to the finished product) manufacturing of the component geometry and the elimination of expensive machining costs incurred in conventional manufacturing. Therefore, there is a strong motivation to develop titanium alloys that can produce a fine, equiaxed grain structure after AM solidification. While there have been several attempts to achieve a columnar to equiaxed transition (CET) in Ti64 by varying AM processing conditions, developing novel titanium alloy compositions that can readily provide fine, equiaxed grains in the as-built condition has been quite promising. A striking example is Ti-Cu binary alloys fabricated using laser powder bed fusion (LPBF) AM, where the ready occurrence of CET resulted in isotropic mechanical properties in the component with mechanical properties exceeding those of wrought Ti64. In order to exploit this approach for AM components with complex geometries, the robustness of CET with respect to the inherent variability in AM thermal conditions for com-

This manuscript has been authored by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725 with the U.S. Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes. The Department of Energy will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan (http://energy.gov/downloads/doe-public-access-plan) This research was sponsored by ExaLearn, an Exascale Computing Project, DOE.

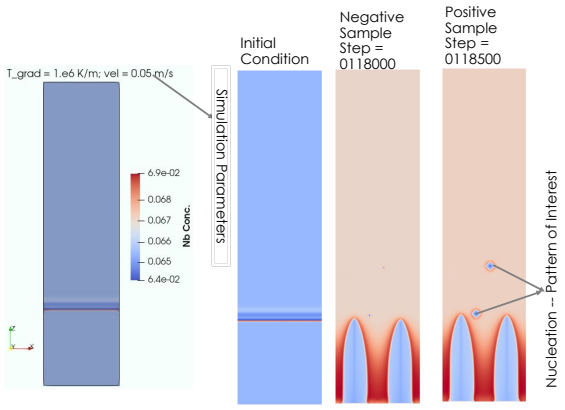



**Figure 1:** PF simulation and the occurrence of nucleation.

plex parts becomes important. Capturing the solidification structure under AM thermal conditions is a complex multi-scale problem and is computationally intensive.

One of the approaches that has been successfully used to simulate the processing-microstructure linkages during AM is the phase field (PF) technique. An important observable of interest in PF simulations is the onset of nucleation in the AM process which occurs for select ranges of the input parameters that are not known a priori. A major goal of PF simulations is to identify these input parameter ranges and understand the inter-relationships that result in nucleation in the AM process. PF simulations are computationally very expensive and often require hundreds and thousands of node hours on state-of-the-art leadership-class computing machines to complete, even for a single simulation. Therefore, sampling even a small region of the multi-dimensional input space to study the onset of nucleation requires the execution of multiple expensive PF simulations incurring tremendous amounts of computing resources. Here, we report the preliminary performance of deep learning (DL) models trained to aid and inform a PF simulation of the likelihood of nucleation occurring in the future (defined by a fixed number of simulation time steps into the future) based on the historical trajectory of an ongoing simulation. The model predictions can be used to stop running simulations intelligently rather than letting them proceed needlessly until the end without the onset of any nucleation. In practice, these predictions optimize overall resource utilization in analyses campaigns requiring multiple computationally expensive PF simulations and ultimately result in tremendous savings of facility-allotted computing node hours and improved user throughput of multi-user supercomputing facilities, the latter being a key quality of service metric for all government- or public-funded multi-user environments.

## II. Related Work

In recent times, machine learning (ML) and deep learning (DL) are being used for many scientific problems predominantly for models such as Improved Forward, Surrogate Models, Reduced-Order, Inverse, and discovering governing equations [1]. In this paper, we are looking at inverse problem of determining simulation parameters from simulation data. There are many works that has focused on building inverse models using DL for experimental data [2]–[4]. One of the primary focuses of this paper is to assist simulation software in reducing unnecessary computations through the use of machine learning models. Similar work in the area of computational fluid dynamics [5] has been discussed in the community. Using image processing based DL models such as convolutional neural network (CNN) has been used in domains such as battery images [6].

While in the current work the focus is mainly on developing a DL model that captures the onset of nucleation based on a set of input parameters for the forward phase field simulations, the focus of a related ongoing work is to develop a surrogate model that predicts the bifurcation of a microstructure evolution path during solid state precipitation [7]. In this work, the phase field simulations are used as input to a Materials Knowledge Systems (MKS) framework [8] that uses statistical descriptions of microstructure using $n$-point correlations [9] along with dimensionality reduction tools based on Principal Component Analysis (PCA) [10], which, in turn, are used in the development of surrogate models for the prediction of microstructure evolution paths based on Gaussian Process Auto-regressive (GPAR) models [11]. Another application of surrogate model development related to solidification of structural alloys under AM conditions is focused on the growth of columnar grains. In this case, the final microstructure is characterized by quantities of interest (QOI) such as grain aspect ratio, grain orientation distribution, grain number, etc. In this case high fidelity phase field simulation data is used as input to a deep neural network to capture the QOIs as a function of thermal conditions during growth [12].

## III. Contributions

This paper reports a novel ML pipeline for the prediction of nucleation in a PF simulation. The pipeline begins by learning a latent feature (or embedding) for every frame of a PF simulation using a supervised DL model. Embeddings over $T$ consecutive time frames are then used by an attention network to predict the likelihood of nucleation after $t < T$ simulation time-steps. Finally, a multi-task DL model is used to simultaneously predict the likelihood of nucleation as a binary classification task as well as predict the simulation input parameters as a regression task. Specific contributions reported in this paper are:

- Robust methods for generation of tokenized PF simulation frames.
- Design of a DL pipeline to predict the occurrence of nucleation in PF simulations.
- Demonstrate that embeddings constructed from our DL models perform better than embeddings constructed

from off-the-shelf convolutional network (LENET-5) for predicting future occurrence of nucleation.
- Demonstrate the efficacy of a novel multitask attention network model in the simultaneous predict of future occurrence of nucleation and the simulation input parameters.

The practical impact of these contributions translate to enormous savings in the utilization of extreme-scale computing resources in typical PF simulation-based analysis campaigns. To our knowledge, such a pipeline has not been reported in the literature before.

The rest of the manuscript is organized as follows. Section IV describes the PF simulation model used to generate the training data. A quick overview of the DL models used here are covered in Section V. The proposed ML pipeline is described in Section VI. Results of our experiments are reported in Section VII ending with our conclusions in Section VIII.

## IV. Data Model and Generation

This work is based on data generated to replicate a typical analysis campaign. It consists of nine representative simulations spanning a combination of two key input parameter values, namely, the local solid-liquid (S-L) interface velocity ($R$) and the temperature gradient ($G$). Each simulation was executed for a varying number of time steps with some resulting in nucleation and others resulting in no nucleation (till the pre-defined end of the simulation). Note that those simulations that ended without any nucleation can potentially still register nucleation if they were executed further into the future. The PF simulations used here were carried out using the MEUMAPPS (Microstructure Evolution Using Massively Parallel Phase-field Simulations) code, described next.

### A. Simulation Model: The MEUMAPS Code

The PF simulations used in this work were performed using the in-house MEUMAPPS code, developed at the Oak Ridge National Laboratory (ORNL) through the Exascale Computing Project (ECP). The PF model used in MEUMAPPS is based on the extension of the KKS binary model developed by Kim et al. [13] to multicomponent alloys [14]. The KKS PF model allows efficient integration with the thermodynamics of multicomponent and concentrated alloys. The equal chemical potential assumption in the KKS model also allows straightforward implementation of the thermodynamically consistent nucleation model described in section 1.2. In the current work, CALPHAD [15] based expressions for the free energies of the solid and liquid phases instead of making use of ideal or regular or dilute solution approximations. An anti-trapping flux term in the concentration evolution equation has been included to eliminate solute-trapping that occurs due to non-equilibrium effects at the s-l interface at high solidification velocities [14]. A Langevin noise is superimposed on the order parameter, to facilitate the destabilization of the s-l interface to form the primary and secondary arms depending on the local thermal conditions and thermodynamic considerations. Detailed description of the PF model used in MEUMAPPS is available in [16]. The final form of the evolution equations for the order parameter and the solute concentration are shown below.

$$\begin{aligned}\frac{\partial \phi}{\partial t} =& M\Bigg(\nabla\sigma^2 \cdot \nabla\phi + \sigma^2\nabla^2\phi \\ &+ \nabla\cdot\left\{\|\nabla\phi\|^2\,\sigma\,\frac{\partial\sigma}{\partial(\nabla\phi)}\right\} - f_\phi\Bigg) + \xi\,,\end{aligned} \tag{1}$$

where $\phi$ is the order parameter, $M$ is the phase field mobility, $\sigma^2$ is the orientation dependent, anisotropic, gradient coefficient, and $\xi$ is the Langevin noise in the order parameter introduced at the $s-l$ interface defined as $0.001 \leq \phi \ll 0.99$. The evolution equation for the concentration field of species $i$ is written as

$$\begin{aligned}\frac{\partial c_i}{\partial t} =& \nabla\cdot\left[h_d(\phi)\;D_i^S\;\nabla c_i^S + \{1-h_d(\phi)\}\;D_i^L\;\nabla c_i^L\;\right] \\ &+ \nabla\cdot\alpha_i\;\frac{\partial\phi}{\partial t}\;\frac{\nabla\phi}{\lfloor\nabla\phi\rfloor}\,,\end{aligned} \tag{2}$$

where $D_i^S$ is the diffusivity of species $i$ in the solid, and $h_d(\phi)$ is an interpolation function, assumed to be equal to $\phi$. The form of the interpolation function $h_d(\phi)$ is based on symmetries that must be imposed to eliminate anomalous interface diffusion and stretching that give rise to solute trapping. The last term on the right-hand side of Eqn. (2) is the anti-trapping flux term where $\alpha_i$ is given by

$$\alpha_i = \frac{\sigma}{\sqrt{2W}}\;(c_i^L - c_i^S)\,.$$

The species concentration at any location $i$ is given by

$$c_i = h_r(\phi)\;c_i^S + [1-h_r(\phi)]\;c_i^L\,.$$

The concentrations $c_i^S$ and $c_i^L$ in the above equation are the solute concentrations at the diffuse interface. In the above equation $h_r(\phi) = h_d(\phi) = \phi$. The assumption of equal chemical potential of the species in the solid and liquid phases leads to an additional set of equations,

$$\frac{\partial G_l}{\partial c_i^L} = \frac{\partial G_s}{\partial c_i^S}$$

The last two equations yield a set of equations that are solved using a Newton-Raphson technique. In general, for a multi-component alloy with $m$ components and $n$ solidifying phases a total of $m \times n$ non-linear system of equations need to be solved at every point in the simulation domain. The calculated interface concentrations are used to solve the evolution of the non-conserved order parameter (Equation (1)) and the conserved solute concentrations (Equation (2)) at

every time step. The evolution equations were solved using a finite-difference scheme using an explicit time integration.

In order to speed up the simulations, the MEUMAPPS code was run on the Summit supercomputer at the Oak Ridge Leadership Computing Facility (OLCF) at Oak Ridge National Laboratory (ORNL) and the Eagle supercomputer at the National Renewable Energy Laboratory (NREL). Parallel implementation was achieved using the Message Passing Interface (MPI) for inter-node communications and OpenACC was used to achieve in-node parallelism using graphical processing units (GPUs).

*1) Thermodynamic Integration:* The chemical free-energy densities (Gibbs energies of phases) are defined using the CALPHAD approach based on the Redlich-Kister polynomials [15] to describe the effect of solute interaction coefficients on the excess free energies of liquid and solid phases. Accordingly, the total free energy of mixing for a phase, $p$, for a binary alloy, $G_m^p$ is given by,

$$\begin{aligned} G_m^p =& c_A\, G_A^0 + c_B\, G_B^0 + RT\, \{c_A \ln c_A + c_B \ln c_B\} \\ &+ c_A c_B \sum_{i=0}^{n_{AB}} L_{A,B;i}\, (c_A - c_B)^i \end{aligned} \tag{3}$$

where the first two terms on the right-hand side represent the standard free energy of formation, the third term represents the ideal entropy of mixing and the last term represents the excess energy defined using the binary interaction coefficients, $L_{A,B;i}$. For the Ni-Mo-Nb alloy used in the simulations, $G_A^0, G_B^0$ and $L_{A,B;i}$ were obtained from the TCNi8 database used in the thermodynamic database, ThermoCalc.

*2) Implementation of the Nucleation Model:* The local thermodynamic driving force, $\nabla G_v$, is calculated based on the composition of the liquid and the solid nucleus composition. The nucleation density is calculated using the classical heterogeneous nucleation theory.

$$I = I_0 exp\left[-\frac{16\pi\sigma^3 f(\theta)}{3kT(\Delta G_v)^2}\right] \tag{4}$$

In Eq. (4), sigma is the solid-liquid interfacial energy and f-theta is a function that is related to the ability of the heterogeneous substrate to wet the liquid.

$$P_n = 1 - exp(-Iv\Delta T) \tag{5}$$

The probability of nucleation is given by Eq. (5). Then Eq. (4) and Eq. (5) are used to calculate the probability of nucleation. A random number generator between 0 and 1.0 is used to declare the site as a solid or not. The solid site is assigned the concentration $C_s$. The excess solute is then distributed uniformly to all the existing liquid sites. This step is an approximation because it assumes just at one time step that the solute diffusivity in the liquid is infinite. However, it serves as an efficient approach to linking the nucleation step with the existing PF framework. In the succeeding time steps, the sharp interface between the nucleus and the surrounding liquid evolve smoothly through the formation of the diffuse interface and the consequent solute enrichment of the surrounding liquid consistent with growth velocity of the nucleus. The degree of heterogeneity in the nucleation is handled by simply varying $f(\theta)$ and $I_\theta$.

### *B. Data Generation*

PF simulations were performed in two-dimensions using a rectangular domain geometry. Zero flux boundary conditions were used for the side faces for the order parameter and the concentration fields. Zero flux boundary conditions were used for the concentration field for the top and bottom faces. The temperature boundary conditions were imposed as follows. The simulation domain initially contained a small layer of solid of thickness, $z_0$, at the bottom and the remaining portion was filled with liquid, both with the average composition of the alloy. The temperature at the s-l interface was set equal to the liquidus temperature of the alloy. The temperature gradients in the solid and liquid were assumed to equal across the s-l interface. The simulation domain was cooled at a rate of $G \times R$ at every time step, where $R$ is the imposed solidification rate.

Since the liquid in contact with the solid is initially at the liquidus temperature of the alloy, there is rapid melting at the s-l interface. Solidification in the positive build direction starts only after the initial rapid melting, followed by a much slower melting during which the s-l interface temperature drops to some temperature between the solidus and liquidus. To prevent the s-l interface from moving out of the computational domain during melting, the domain is initially translated in the downward direction with the velocity of the s-l interface to keep the s-l interface position stationary in the moving frame. Similarly, during solidification in the positive build direction, the domain is translated in the upward direction with the velocity of the s-l interface. This feature allows the detailed investigation of the dynamics of the s-l interface during the initial transient.

Typical parameters used in the PF simulations are shown in Table I. The diffusion coefficient of substitutional solutes in most metallic liquids is $10^{-9}\ m^2/s$. A value of $2 \times 10^{-9}$ $m^2/s$ is used in the current simulations. The s-l interfacial energy during dendritic/cellular solidification is unknown for this study's Ni-Mo-Nb alloy. An approximate value of 0.16 $J/m^2$ is assumed. The current formulation of the PF model does not allow solute diffusion in the solid.

For $R$ 0.05 $m/s$ and 0.08 $m/s$, simulations are run for different $G$ values ($10^4, 10^5, 10^6, 10^7$ and $10^8 K/m$) to yield simulations which are stored as VTK images [17] at different time resolutions and varying sizes. We use the VTK Python interface to read every frame and scaled them to Portable Network Graphics (PNG) image of size $1000 \times 300$ pixels. These images are then used to train the three models, viz.,

**Table I:** Simulation Parameters

| Parameter | Input |
|---|---|
| Mesh resolution | $5 \times 10^{-9}$ $m$ or $10^{-8}$ $m$ |
| Simulation domain Size | $500 \times 2 \times 1500$ |
| Time step | $1 \times 10^{-9}$ $s$ or $5 \times 10^{-9}$ $s$ |
| Alloy | Ti-06Cu (atom fraction) |
| Anisotropy s-l interfacial energy | 0.05 |
| s-l interface energy | 0.16 $J/m^2$ |
| Nucleation parameters - Homogeneous | $I_0 = 1 \times 10^{39} m^{-3} s^{-1}, f(\theta) = 0.999$ |
| Nucleation parameters - Heterogeneous | $I_0 = 1 \times 10^{30} m^{-3} s^{-1}, f(\theta) = 0.006$ |
| Diffusion coefficient in liquid | $2 \times 10^{-9} m^2/s$ |
| Diffusion coefficient in solid | 0.0 |
| Thermal conditions | $R(m/s) = 0.05$ or $0.08$; $G(K/m) = 10^4$–$10^8$ |
| Initial undercooling at s-l interface | 0K (s-l interface at the liquidus temperature of the alloy) |

N-CNN, N-ATTN and N-ATTN-MT (see Fig. 2). When training the multi-task N-ATTN-MT model, the $R$ and $G$ regression labels are transformed to bring their errors closer.

## V. Deep Learning Models

This section provides a brief overview of the architectures of the neural network models used to the design the DNN models in this study.

### A. Convolutional Neural Network

A convolutional neural network (CNN) [18] is a type of feed-forward neural network composed of alternate sequences of convolution and pooling layers. Each node $n$ in a convolutional layer $i$ computes the following operation:

$$y_n^i = \sigma_{i,n} \left( \sum_m^{M^i} \mathbf{w}_{n,m}^i * \mathbf{x}_m^i + b_n^i \right)$$

where $\mathbf{x}_m^i$ represents the $m$-th input map at layer $i$; $\mathbf{w}_{n,m}^i$, $b_n^i$ and $\sigma_{i,n}$ represent node parameters, namely, the $m$-th convolutional kernel, the bias and the activation function, respectively; $M^i$ are the number of input maps at layer $i$ (which correspond to the output maps at layer $i-1$, i.e. the number of nodes at layer $i-1$); and $*$ denotes a convolutional operation. The activation function is usually a rectified linear unit (ReLU) which corresponds to the following operation $\mathrm{ReLU}(\nu) = \max(0, \nu)$.

The convolutional layers reduce the number of free parameters (compared with regular densely connected neural networks) since they learn compactly supported kernels (typically $3 \times 3$ or $5 \times 5$) that are globally optimal throughout the input signal and are able to capture local invariant patterns in the data. This makes them very successful models for image processing tasks, requiring minimal pre-processing steps. The pooling layers sweep through the input image applying a reduction operation (average or maximum) over a fixed neighborhood structure. This reduces the dimensionality of the image being processed while making the model more robust to local variations in the input.

The parameters of the CNN networks are determined by minimizing a loss function over the training data set. The loss function is selected according to the specific task that the model is performing, usually cross-entropy loss for classification problems or a mean square error (MSE) for regression problems.

### B. Attention Networks

Attention networks (ATTN) are deep learning architectures that make use of the attention mechanism [19]. The attention mechanism treats the input as a collection of data tokens and computes (attention) scores between pairs of tokens trying to estimate and understand their reciprocal influence. This allows to consider each token in the context of all the other tokens, which effectively captures long-range interactions and complex dependencies, escaping from the myopic view of other models that rely only on local correlations.

The computations in ATTN can be described by the following equations [20]. Scores between pairs of tokens $\mathbf{x}_i \in \mathbb{R}^d$ and $\mathbf{x}_j \in \mathbb{R}^d$ are estimated as

$$\alpha_{i,j}^{(h)} = \mathrm{softmax}_j \left( \frac{\langle Q^{(h)}(\mathbf{x}_i), K^{(h)}(\mathbf{x}_j) \rangle}{\sqrt{k}} \right),$$

where query $Q^{(h)}(\mathbf{x}_i) = W_{h,q}^T \mathbf{x}_i$ and key $K^{(h)}(\mathbf{x}_i) = W_{h,k}^T \mathbf{x}_i$ and with $W_{h,q} \in \mathbb{R}^{d \times k}$ and $W_{h,k} \in \mathbb{R}^{d \times k}$ layer parameters, and $\mathrm{softmax}(\mathbf{x})_\nu = \exp(x_\nu) / \sum_{\lambda=1}^{\Lambda} \exp(x_\lambda)$.

The output $\mathbf{z}_i \in \mathbb{R}^d$ of the multi-headed self-attention layer $i$ is then computed via layer normalization, residual connections,

$$\begin{aligned} \mathbf{u}_i' &= \sum_{h=1}^{H} W_{c,h}^T \sum_{j=1}^{n} \alpha_{i,j}^{(h)} V^{(h)}(\mathbf{x}_j), \\ \mathbf{u}_i &= \mathrm{LayerNorm}(\mathbf{x}_i + \mathbf{u}_i'; \gamma_1, \beta_1), \end{aligned}$$

where value $V^{(h)}(\mathbf{x}_i) = W_{h,\nu}^T \mathbf{x}_i$, and with $W_{h,\nu} \in \mathbb{R}^{d \times k}$, $W_{c,h} \in \mathbb{R}^{k \times d}$, and $\gamma_1, \beta_1 \in \mathbb{R}^d$ layer parameters, and a fully connected layer

$$\begin{aligned} \mathbf{z}_i' &= W_2^T \, \mathrm{ReLU}(W_1^T \mathbf{u}_i), \\ \mathbf{z}_i &= \mathrm{LayerNorm}(\mathbf{u}_i + \mathbf{z}_i'; \gamma_2, \beta_2), \end{aligned}$$

with $W_1 \in \mathbb{R}^{d \times m}$, $W_2 \in \mathbb{R}^{m \times d}$, and $\gamma_2, \beta_2 \in \mathbb{R}^d$ layer parameters, and with the LayerNorm function

$$\mathrm{LayerNorm}(\mathbf{z}; \gamma, \beta) = \gamma \frac{\mathbf{z} - \mu_\mathbf{z}}{\sigma_\mathbf{z}} + \beta, \quad \gamma, \beta \in \mathbb{R}^k,$$

$$\mu_\mathbf{z} = \frac{1}{k} \sum_{i=1}^{k} \mathbf{z}_i, \qquad \sigma_\mathbf{z} = \sqrt{\frac{1}{k} \sum_{i=1}^{k} (\mathbf{z}_i - \mu_\mathbf{z})^2}.$$

Again, as any other neural network model, the ATTN network is trained to produce a desired output by minimizing a loss function over the training data set. The minimization determines a set of model parameters, corresponding to

weight matrices $W$ and LayerNorm $\gamma$, $\beta$ parameters, that better approximate the desired output. The hyper-parameters are the number of tokens per input $n$ (e.g. number of image patches or length of data sequence), the dimension of the tokens $d$ (i.e. number of features, e.g. size of each patch), the length of the key embedding $k$, the dimension of the fully connected layer $m$, the number of self-attention heads $H$, and the number of self-attention layers $L$.

## VI. Deep learning Models for Nucleation Prediction

In this section, we discuss our deep learning models for predicting the likelihood of the occurrence of nucleation in a future simulation time-step based on information collected over a finite number of past time-steps of a running simulation. We are particularly interested in attention networks, since they are effective models for capturing long-range interactions, a feature that could be extremely useful for predicting the onset of nucleation which may or may not occur at a distant future time-step, depending on complex relationships between thermal conditions and the different physical variables evolving in the AM system.

To get better insights into the process, we break the learning into a set of related sub-tasks. In the first sub-task, we explore how to produce good inputs for the ATTN. In the second sub-task, we explore the prediction of the nucleation onset on a fixed future time-step using an ATTN. In the third sub-task, we tackle the problem of simultaneous prediction of nucleation onset and associated simulation parameters ($R$ and $G$) with an ATTN. Each of these sub-tasks is described in more detail next.

### A. Embedding for ATTN

One of the fundamental design decisions to make regarding applying an ATTN to a continuous problem is how to pre-process the data available to build a meaningful embedding (i.e. a discrete tokenization) that can be used as an input to the attention network. In the case of estimating the likelihood of the onset of nucleation in AM, it seems practical to work directly with the images representing the frame by frame evolution of the solidification microstructures. One can think on tokenizing each of the individual $1000 \times 300$ frames produced by the simulation via non-overlapping image patches. However, we may loose important information by disregarding the neighborhoods in the borders of the patches, while going in the direction of constructing overlapping patches, yields an unwieldy representation intractable from the point of view of attention networks that compute pairwise correlations between tokens. Since CNNs are efficient models to capture local correlations, and have been shown to be highly successful in image processing workflows, we decide in favor of using a CNN as the model to learn a good embedding of the individual simulation frames. Of course, the problem now becomes how to characterize what a *good embedding is*. For this purpose, we pose an auxiliary and related supervised binary classification problem: we consider every frame as (partially) independent and label each frame $i$ at time $t_i$ as 0 if there is no nucleation onset on $t \in [0, t_i]$ or 1 otherwise. We construct a CNN, denoted as N-CNN (see Figure 2a), that finishes in two fully connected layers, where the first fully connected layer produces a latent feature embedding of size $k$ and the second fully connected one projects the size $k$ latent feature into two outputs for carrying out the binary classification. Training the N-CNN for this binary classification setup provides a $k$ size embedding for each simulation frame and an objective metric to evaluate its quality.

### B. Prediction of Nucleation Onset

We formulate the problem of predicting the likelihood of the occurrence of nucleation in a future simulation time-step as a binary classification problem. Specifically, we use as input a historically time-window of $T$ past time-steps in the simulation and aim to predict if the nucleation onset occurs at a fixed future simulation time-step $\tau$ or not. In other words, given $T = 50$ frames of a simulation from $t_1$ to $t_{50}$ and $\tau = 25$, we determine if the nucleation happened on the 75th step $t_{75}$ or not. Analogously, for input frames from $t_2$ to $t_{51}$, we aim to predict if the onset of nucleation happened on the 76th step $t_{76}$ or not, and so on. Note that the greater the $\tau$ the more savings in computation, but the more difficult the prediction problem.

For this task, we use an ATTN to collect information of the sequence of $T$ past time-step frames and hope to exploit the self-attention mechanism to capture long-range interactions enough to predict if the onset of nucleation happens at a time-step $\tau$ in the future or not. We apply the N-CNN to embed each and every frame of the $T$ frames of a simulation into a $k$ length vector and provide this as an input to a multihead self-attention layer with $h$ heads. The number of $h$ heads are chosen such that it can exactly divide $k$. We obtain the average of the multiple heads after applying layer-wise normalization on the output of multi-head attention. We use this averaged vector for determining if the nucleation occurred or not. Thus, this multi head self-attention network, denoted by N-ATTN (see Figure 2b), predicts the occurrence or not of nucleation in a future simulation time-step $\tau$ based on information collected over $T$ past time-steps of a running simulation.

### C. Multi-Task Prediction of Nucleation Onset and Simulation Parameters

We move on to the final step of determining the simulation parameters – $R$ and $G$– along with predicting the occurrence of nucleation. For this goal, we construct a multi-task model, called Multi-task Nucleation Attention Network and denoted as N-ATTN-MT (see Figure 2c), which simultaneously

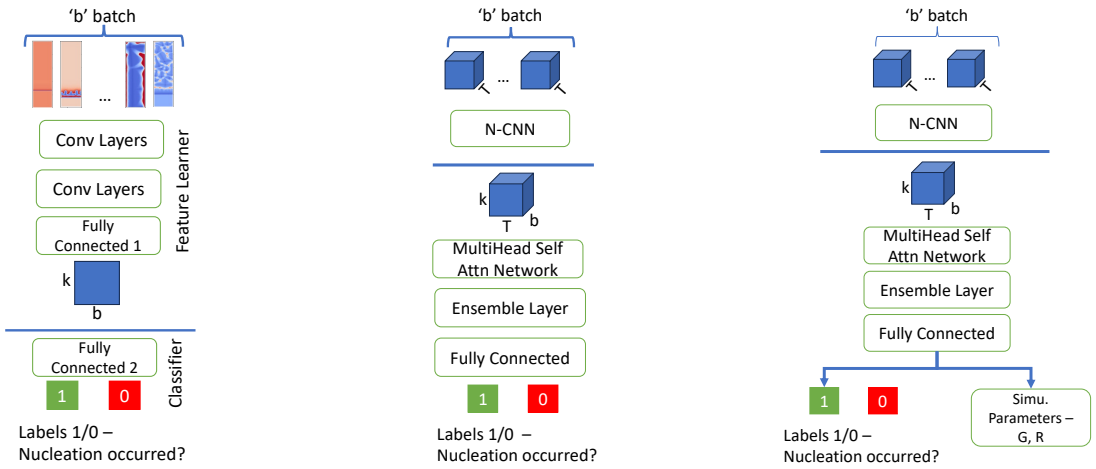



**Figure 2:** (a) Nucleation Convolution Neural Network (N-CNN). (b) Nucleation Attention Network (N-ATTN). (c) Multi-task Nucleation Attention Network (N-ATTN-MT).

solves the binary classification problem of Section VI-B and the regression problem of predicting $R$ and $G$. Note that this model also exploits the embeddings determined by the N-CNN.

For the total loss function we consider the sum of both classification and regression losses. In order to appropriately weight both of them, we use a regularization parameter $\lambda > 0$ that scales classification error with $\lambda$ and regression error as $1 - \lambda$. Moreover, in-order to guarantee that the errors are similar in magnitude across classification and regression tasks, we transform the regression labels for the simulation parameters. For regression, we use mean squared error (MSE) as the loss function. We have to chose an appropriate classification loss function that returns values in a similar range to MSE, otherwise, the performance of one task will dominate over the other. In our case, we chose negative log likelihood loss along with MSE for our multitask network N-ATTN-MT.

Details about network architectures and training are discussed in the next section.

## VII. Experiments

This section describes the training of the different DL models and compare their performance. We are able to demonstrate the superiority of the embeddings from our proposed model N-CNN over off-the-shelf LeNet-5 [21] for predicting the occurrence of nucleation and determining the simulation parameters ($R$ and $G$) in phase field simulations. We begin by explaining the computing environment, baselines, network architectures and metrics. Finally, we present the training details of our models, report the results and present the observations of our experiments.

### A. Computing Environment

The experiments were performed on an Ubuntu 22.03 Linux server with 3 NVIDIA GP104GL Quadro P5000 GPUs. All the models were developed with PyTorch 2.0 and the Cuda 12.0 toolkit.

### B. Baselines

We compare our proposed N-CNN against the popular off-the-shelf model LeNet-5 [21]. The classification and the regression tasks of our N-ATTN and N-ATTN-MT are compared with the embeddings obtained from the LeNet-5 model. The size of the first fully connected layer of default LeNet-5 model was modified to reflect the $1000 \times 300$ of our simulation data.

### C. Network Architectures and Training

We build the N-CNN model using three convolution layers. In the first layer, we use 10 filters each of size $11 \times 11$. Both of the rest of the two convolution layers have a kernel size of $5 \times 5$ and use 20 and 32 filters, respectively. In between the convolution layers, we use a $2 \times 2$ neighborhood max pooling and ReLU activation function. This model ends with two fully connected layers. The hyperparameters, such as the number of convolution filters, size of the filters, and latent feature size, were determined through GridSearch [22].

The N-CNN and LeNet-5 were trained using 9 different simulations, corresponding to a total of 919 frames with 397 positive samples where the nucleation has occurred and 522 negative samples without nucleation. We use a random 20% of the total samples as hold out set for testing. For training the N-CNN, we used Adam optimizer with learning rate 1e-4 and momentum of 0.5. LeNet-5 was trained with SGD optimizer with a learning rate of le-6 and momentum of 0.3. Both trainings were run for 50 epochs with 64 samples on every batch.

For the N-ATTN architecture, we use the embedding obtained out of the previous-to-last fully connected (FC) layers

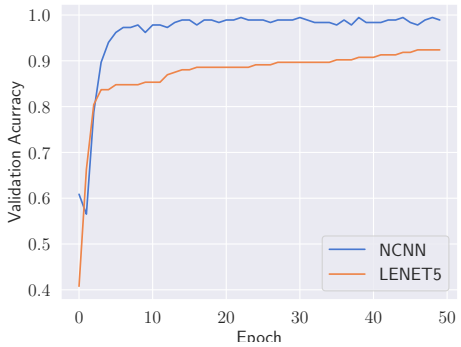



**Figure 3:** Testing accuracy comparison between LENET-5 and N-CNN over 50 epochs. Higher is better.

of N-CNN and LENET-5 models and train two different N-ATTN networks. The FC from N-CNN produced an embedding length of 50 while the FC2 of LENET-5 generated a 84 length embedding. The number of heads for multihead must exactly divide the embedding length. Hence, we used 10 multiheads for N-ATTN for training with embeddings from N-CNN and 12 for embeddings from LENET-5. Both the variants of N-ATTN with embeddings from N-CNN and LENET-5 were trained for 200 epochs with Adam optimizer using a learning rate of 1e-3 momentum of 0.5 and a batch size of 64.

To train the two N-ATTN described architectures, we took a subset of 50 steps from the available 9 different simulations, resulting in 1199 samples. We labeled each of them as positive if the nucleation occurred at $\tau = 25$ time-steps into the future, yielding 742 positive samples – where the nucleation appeared after 25 time steps – and 457 negative samples where it did not. Out of 1199 samples, we used 20% as hold out samples for testing.

For the N-ATTN-MT model, we use the same architecture and training settings (i.e. optimizer, optimization parameters and batch size) from the N-ATTN. We also use the same data, except that we add the two simulation parameters $R$ and $G$ for the regression task. From Table I these take values $R$: 0.05 and 0.08 m/s, and $G$: 1e4, 1e5, 1e6, 1e7, 1e8 K/m. We take a log10 transformation on $G$ values and scaled the $R$ by 100. With these transformations the scales of parameters to predict become comparable.

Note that the total number of parameters for all the trained DL architectures is included in Table II.

### D. Metrics

For comparing the performance of the models on classification tasks, we use accuracy and estimate the confusion matrix for the binary classification. We report the root mean square error (RMSE) for the regression task. For accuracy, the higher the better, while for RMSE the lower the better.

**Table II:** Total Number of Learning Parameters from different DL models. The number of parameters of the N-ATTN and N-ATTN-MT does not include the parameters of embedding models N-CNN or LENET-5.

| Model | Number of Parameters |
|---|---|
| LENET-5 | 34158800 |
| N-CNN | 6362054 |
| N-ATTN w/ N-CNN embeddings | 17186 |
| N-ATTN w/ LENET-5 embeddings | 35682 |
| N-ATTN-MT w/ N-CNN embeddings | 17444 |
| N-ATTN-MT w/ LENET-5 embeddings | 35940 |

### E. Results

*1) Embeddings for nucleation prediction:* We compared our model N-CNN against LENET-5 [21] for building the latent features for every simulation frame. Figure 3 shows the validation accuracy after every epoch. N-CNN converged faster and better over the baseline LENET-5. We used the trained model to understand the accuracy on the entire dataset of 919 samples. The models N-CNN and LENET-5 obtained 99% and 90% respectively. By comparing the Figure 7a with Figure 8a, we can observe that LENET-5 reported more false-negatives over our proposed N-CNN.

*2) Predicting nucleation onset from simulations' past time-steps:* Unlike the N-CNN, the N-ATTN relaxes the assumption of independence of frames and models the temporal relations using multihead self-attention. We used $T = 50$ frames of every simulation to determine if the nucleation occurred after $\tau = 25$ time-steps into the future. Figure 4, reports accuracy on the hold out set after every epoch. We can observe that N-ATTN with embeddings from N-CNN performed much better over LENET-5. Despite N-ATTN with embeddings from LENET-5 having more parameters (see Table II), it converged around a testing accuracy of 65%, while N-ATTN with embeddings from N-CNN, converged to a testing accuracy around 98%.

Figures 7b and 8b help us further understand the performance of the models. As in the N-CNN case, we can observe more false positives in N-ATTN with embeddings from LENET-5. Thus our fine tuned N-CNN produces better quality embeddings over LENET-5 which contributes to a more accurate detection of nucleation even at a distance of $\tau$ future time-steps in the simulation.

*3) Multitask Prediction:* We conducted experiments to understand the ability of DL to find the simulation parameters along with the occurrence of nucleation on a simulation. The accuracy obtained is displayed in Figure 5. This shows similar characteristics as the Figure 4 and points towards the fact that the embeddings from the N-CNN are better than the embeddings from LENET-5 even while training in multitask setting. However, from Figure 6, the RMSE from the LENET-5 embeddings is slightly better over N-CNN embeddings. Also, we can see that the multitask training with N-CNN embeddings reduced the number of false positives from Figure 8c.

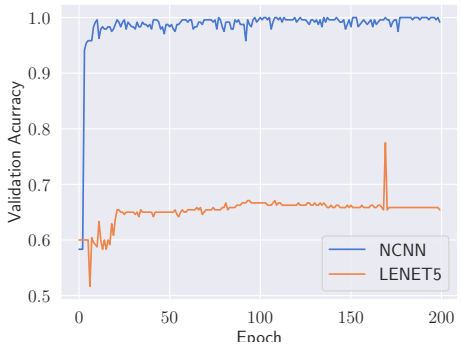



**Figure 4:** Testing accuracy comparison for N-ATTN with embeddings obtained from LENET-5 and N-CNN over 200 epochs. (Higher is better)

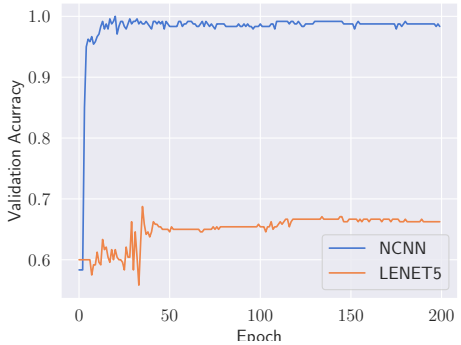



**Figure 5:** Testing accuracy comparison for N-ATTN-MT with embeddings obtained from LENET-5 and N-CNN over 200 epochs. (Higher is better).

## VIII. CONCLUSIONS AND FUTURE WORK

In this paper, we present the design, implementation and performance of a carefully designed ML pipeline with the primary goal of identifying those PF simulations that can be stopped prematurely to avoid needless computations that do not culminate in producing the observables of interest - occurrences of nucleation. Avoiding these simulations optimizes large-scale workloads of PF simulations and ultimately results in enormous savings in terms of computing node-hours and improves user-throughput of these facilities, both of which are important considerations for government-funded multi-user facilities such as the Oak Ridge Leadership Computing Facility. The pipeline first uses a custom-made DL model called N-CNN to learn the embeddings from individual time frames of the PF simulations and uses them with a self-attention network called N-ATTN to predict the future occurrence of nucleation as a binary classification task. A multi-task model called the N-ATTN-MT is then trained to perform the preceding classification task simultaneously with the prediction of the simulation input parameters as a concurrent regression task. Performance of the ML pipeline is tested using data generated by the massively parallel MEUMAPPS PF simulation code developed within the ExaAM ECP Project. The results conclusively show that the performance of the proposed learning pipeline using *smaller* custom-made DL models outperforms the baseline performance in which a *larger* off-the-shelf LENET-5 model is used to learn the same embeddings. Ongoing and future work aims to reduce false-positives and false-negatives by interpreting the learned weights of our models. The models will be further refined by overcoming data paucity stemming from expensive PF simulations using a combination of data augmentation techniques and additional scientific information like orientation maps.

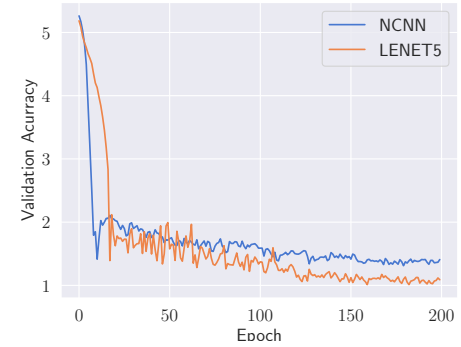



**Figure 6:** Testing RMSE comparison of determining simulation parameters $R$ and $G$ using N-ATTN-MT. The plot compares results with embedding from LENET-5 and N-CNN over 200 epochs. (Lower is better).

## REFERENCES

[1] A. Karpatne, R. Kannan, and V. Kumar, "Knowledge guided machine learning: Accelerating discovery using scientific knowledge and data," 2022.

[2] C. Garcia-Cardona, R. Kannan, T. Johnston, T. Proffen, K. Page, and S. K. Seal, "Learning to predict material structure from neutron scattering data," in *2019 IEEE International Conference on Big Data (Big Data)*. IEEE, 2019, pp. 4490–4497.

[3] C. Garcia-Cardona, R. Kannan, T. Johnston, T. Proffen, and S. K. Seal, "Structure prediction from scattering profiles: A neutron-scattering use-case," in *Knowledge Guided Machine Learning*. Chapman and Hall/CRC, 2022, pp. 287–304.

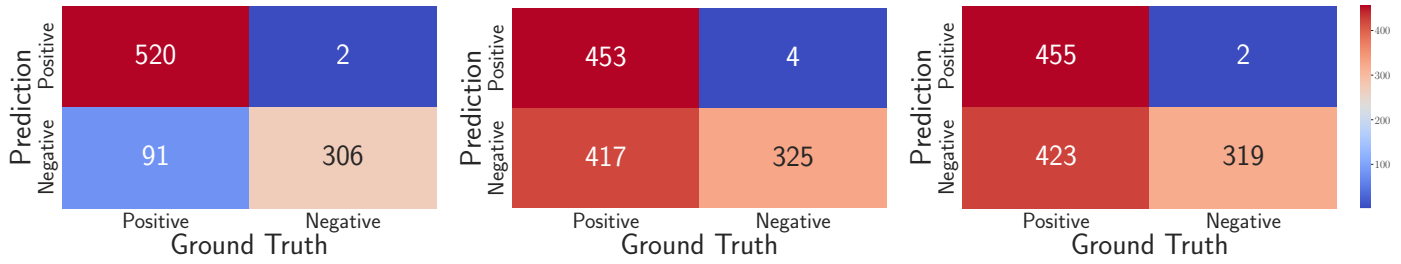
[4] ——, "Structure prediction from neutron scattering profiles: A data sciences approach," in *2020 IEEE International Conference on Big Data (Big Data)*. IEEE, 2020, pp. 1147–1155.

[5] R. Bhaskaran, R. Kannan, B. Barr, and S. Priebe, "Science-guided machine learning for wall-modeled large eddy simulation," in *2021 IEEE International Conference on Big Data (Big Data)*. IEEE, 2021, pp. 1809–1816.

[6] A. Tabassum, N. Muralidhar, R. Kannan, and S. Allu, "Matphase: Material phase prediction for li-ion battery reconstruction using hierarchical curriculum learning," in *2022 IEEE International Conference on Big Data (Big Data)*. IEEE, 2022, pp. 1936–1941.

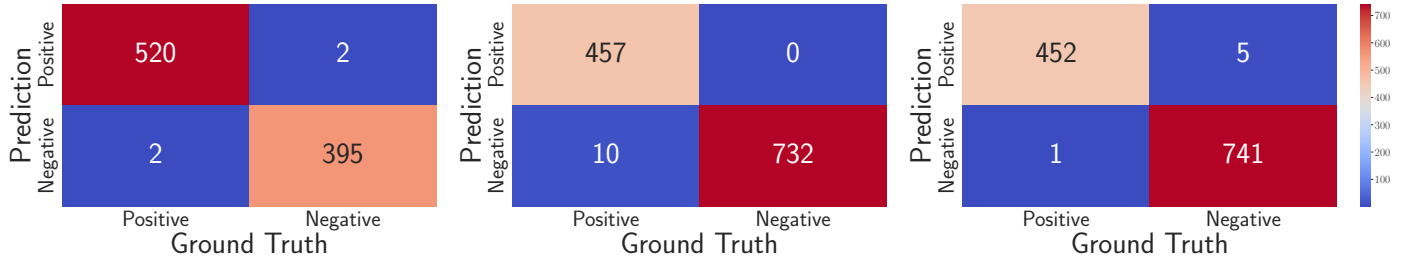
[7] A. Marshall, S. R. Kalidindi, and B. Radhakrishnan, "A Gaussian process auto-regressive model for capturing microstructure evolution paths in a Ni-Mo-Nb alloy," (submitted).

[8] S. Kalidindi, *Hierarchical materials informatics: novel analytics for materials data*. Elsevier, 2015.




**(a)** **(b)** **(c)**

**Figure 7:** Confusion matrices corresponding to (a) LeNet-5 (b) N-ATTN (c) N-ATTN-MT with LeNet-5 embeddings.




**(a)** **(b)** **(c)**

**Figure 8:** Confusion matrices corresponding to (a) N-CNN (b) N-ATTN (c) N-ATTN-MT with N-CNN embeddings.


[9] A. Cecen, Y. Yabansu, and S. Kalidindi, "A new framework for rotationally invariant two-point spatial correlations in microstructure datasets," *Acta Materialia*, vol. 158, no. 53, 2018.

[10] C. Suh, A. Rajagopalan, X. Li, and K. Rajan, "The application of principal component analysis to materials science data," *Data Science Journal*, vol. 1, no. 19, 2002.

[11] Y. Yabansu, V. Rehn, J. Hötzer, B. Nestler, and S. Kalidindi, "Application of gaussian process autoregressive models for capturing the time evolution of microstructure statistics from phase-field simulations for sintering of polycrystalline ceramics," *Modelling and Simulation in Materials Science and Engineering*, vol. 27, no. 084006, 2019.

[12] Y. Qin, S. DeWitt, B. Radhakrishnan, and G. Biros, "GrainNN: A neighbor aware long short-term memory network for predicting microstructure evolution during polycrystalline formation," *Computational Materials Science*, vol. 218, no. 111927, 2023.

[13] S. Kim, W. Kim, and T. Suzuki, "Phase-field model for binary alloys," *Phys. Rev. E*, no. 60, pp. 7186–7197, 1999.

[14] S. G. Kim, "A phase-field model with antitrapping current for multicomponent alloys with arbitrary thermodynamic properties," *Acta Materialia*, vol. 55, no. 13, pp. 4391–4399, 2007.

[15] U. R. Kattner, "The CALPHAD method and its role in material and process development," *Technol. Metal. Mater. Miner.*, vol. 13, pp. 3–15, 2016.

[16] B. Radhakrishnan, S. B. Gorti, J. A. Turner, R. Acharya, J. A. Sharon, A. Staroselsky, and T. El-Wardany, "Phase field simulations of microstructure evolution in IN718 using a surrogate Ni–Fe–Nb alloy during laser powder bed fusion," *Metals*, vol. 9, no. 1, p. 14, 2018.

[17] L. S. Avila, U. Ayachit *et al.*, *The VTK User's Guide*, 11th ed. Kitware, Inc., 2010. [Online]. Available: https://www.kitware.com/products/books/VTKUsersGuide.pdf

[18] Y. LeCun, Y. Bengio, and G. E. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015.

[19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *31st Conference on Neural Information Processing Systems (NIPS 2017)*, 2017.

[20] J. Thickstun, "The transformer model in equations." [Online]. Available: https://johnthickstun.com/docs/transformers.pdf

[21] Y. LeCun *et al.*, "Lenet-5, convolutional neural networks," *URL: http://yann. lecun. com/exdb/lenet*, vol. 20, no. 5, p. 14, 2015.

[22] A. Tam, "How to Grid Search Hyperparameters for PyTorch Models - MachineLearningMastery.com — machinelearningmastery.com," https://machinelearningmastery.com/how-to-grid-search-hyperparameters-for-pytorch-models/, [Accessed 04-09-2023].