UNIVERSITY OF CALIFORNIA

Los Angeles

Machine Learning Modeling with Application to Laser Powder Bed Fusion Additive Manufacturing Process

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy
in Chemical Engineering

by

Yi Ming Ren

2022

ABSTRACT OF THE DISSERTATION

Machine Learning Modeling with Application to Laser Powder Bed Fusion Additive Manufacturing Process

by

Yi Ming Ren

Doctor of Philosophy in Chemical Engineering

University of California, Los Angeles, 2022

Professor Panagiotis D. Christofides, Chair

Big data plays an important role in the fourth industrial revolution, which requires engineers and computers to fully utilize data to make smart decisions to optimize industrial processes. In the additive manufacturing (AM) industry, laser powder bed fusion (LPBF) and direct metal laser solidification (DMLS) have been receiving increasing interest in research because of their outstanding performance in producing mechanical parts with ultra-high precision and variable geometries. However, due to the thermal and mechanical complexity of these processes, printing failures are often encountered, resulting in defective parts and even destructive damage to the printing platform. For example, heating anomalies can result in thermal and mechanical stress on the building part and eventually lead to physical problems such as keyholing and lack of fusion. Many of the aforementioned process errors occur during the layer-to-layer printing process, which makes in-situ process monitoring and quality control extremely important. Although in-situ sensors are extensively developed to investigate and record information from the real-time printing process, the lack of efficient in-situ defect detection techniques specialized for AM processes makes real-time process monitoring and data analysis extremely difficult. Therefore, to help process engineers analyze sensor information and efficiently filter monitoring data for transport and storage, machine learning and data processing algorithms are often implemented. These algorithms integrate the functionality of automated data processing, transferring, and analytics. In particular, sensor data often takes the form of images, and thus, a prominent approach to conducting image analytics is through the use of convolutional neural networks (CNN). Nevertheless, the industrial utilization of machine learning methods often encounters problems such as limited and biased training datasets. Hence, simulation methods, such as the finite-element method (FEM), are used to augment and improve the training of the deep learning process monitoring algorithm.

Motivated by the above considerations, this dissertation presents the use of machine learning techniques in process monitoring, data analytics, and data transfer for additive manufacturing processes. The background, motivation, and organization of this dissertation are first presented in the Introduction chapter. Then, the use of FEM to model and replicate in-situ sensor data is presented, followed by the use of machine learning techniques to conduct real-time process monitoring trained from a mixture of experimental and replicated sensor image data. In particular, a cross-validation algorithm is developed through the exploitation of different sensor advantages and is integrated into the machine learning-assisted process monitoring algorithm. Next, an application of machine learning (ML) to non-image sensor data is presented as a neural network model that is developed to estimate in-situ powder thickness to account for recoater arm interactions. Subsequently, an integrated AM smart manufacturing framework is proposed which connects the different manufacturing hierarchies, particularly at the local machine, factory, and cloud level. Finally, in addition to the AM industry, the use of machine learning, specifically neural networks, in model predictive control (MPC) for dynamic nonlinear processes is reviewed and discussed.

The dissertation of Yi Ming Ren is approved.

Contents

List of Figures

List of Tables

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my advisor, Professor Panagiotis D. Christofides, for his support and encouragement of both my technical work and life goals. Professor Christofides sets an example of excellence as a researcher, mentor, instructor, and role model. His mentoring on control theory, process/systems engineering, and most importantly life decisions has made a profound impact on both my character and career. I would also like to thank Professor Samanvaya Srivastava, Professor Dante A. Simonetti, and Professor Tsu-Chin Tsao for serving on my doctoral committee. In addition, I would like to thank all of my colleagues with whom I have worked with over the years in the Christofides research group, including Dr. Zhihao Zhang, Dr. Scarlett Chen, Fahim Abdullah, Sungil Yun, and Aisha Alnajdi. I would particularly like to thank Professor Zhe Wu, Dr. Yangyao Ding, Dr. Yichi Zhang, Mohammed Alhajeri, and Junwei Luo with whom I have collaborated extensively and spent long hours working on papers together.

I am also grateful for my family and friends and for their encouragement, support, and patience throughout my graduate career. In particular, I am grateful for my parents, Wen Cheng and Tianli Ren. In addition, I am also thankful toward my girlfriend, Kaili Ding, and my friends, Tyler Han, Willis Peng, Albert Lin, Xingrong Qian, Xinghong Tang, Sihan Wang, Guangning Yang, Jiahao Zhang, Dr. Kan Yao, Dr. Weiqi Zhang, and Dr. Qiyao Zhu, for keeping me motivated. Most importantly, I would like to thank my grandparents, Jianli Liu and Biwei Cheng, the latter who recently passed away but will always be in my heart. They have both molded me into the person I am today and have given me the strength and determination to become fellow doctors following their footsteps.

Financial support from UCLA Graduate Division, the US Department of Energy (DOE), and the US National Science Foundation (NSF) is gratefully acknowledged, and my work could not have been done without this support.

Chapter 2 contains versions of: Ren Y., Y. Zhang, Y. Ding, Y. Wang and P. D. Christofides, "Computational Fluid Dynamics-Based in-Situ Sensor Analytics of Direct Metal

Laser Solidification Process Using Machine Learning" *Computers & Chemical Engineering*, 143, 107069, 2020.

Chapter 3 contains versions of: Ren Y., Y. Zhang, Y. Ding, T. Liu, C. S. Lough, M. C. Leu, E. C. Kinzel, and P. D. Christofides, "Finite Element Modeling of Direct Metal Laser Solidification Process: Sensor Data Replication and Use In Defect Detection and Data Reduction via Machine Learning" *Chem. Eng. Res. & Des.*, 171, 254-267, 2021.

Chapter 4 contains versions of: Liu, T., C. S. Lough, H. Sehhat, Y. M. Ren, P. D. Christofides, E. C. Kinzel, and M. C. Leu, "In-Situ Thermographic Inspection for Local Powder Layer Thickness Estimation in Laser Powder Bed Fusion," *Additive Manufacturing*, 55, 102873, 2022.

Chapter 5 contains versions of: Ren, Y., Y. Ding, Y. Zhang, and P. D. Christofides, "A Three-Layer Hierarchical Framework for Additive Manufacturing," *Digit. Chem. Eng.*, 1, 100001, 2021.

Chapter 6 contains versions of: Ren, Y., M. S. Alhajeri, J. Luo, S. Chen, F. Abdullah, Z. Wu, and P. D. Christofides, "A Tutorial Review of Neural Network Modeling Approaches for Model Predictive Control," *Computers & Chemical Engineering*, in press.

Curriculum Vitae

Education

University of California, Los Angeles **Sep. 2018 - June 2022**

Ph.D., Chemical Engineering **Los Angeles, CA**

Advisor: Professor Panagiotis D. Christofides

University of California, Los Angeles **Sep. 2014 - June 2018**

B.S., Chemical and Biomolecular Engineering **Los Angeles, CA**

Specialization: semiconductor manufacturing

Journal Publications

1. **Ren Y.**, M. S. Alhajeri, J. Luo, S. Chen, F. Abdullah, Z. Wu, and P. D. Christofides, "A Tutorial Review of Neural Network Modeling Approaches for Model Predictive Control," *Computers & Chemical Engineering*, in press.

2. **Ren Y.**, Y. Zhang, Y. Ding, T. Liu, C. S. Lough, M. C. Leu, E. C. Kinzel, and P. D. Christofides, "Finite Element Modeling of Direct Metal Laser Solidification Process: Sensor Data Replication and Use In Defect Detection and Data Reduction via Machine Learning" *Chem. Eng. Res. & Des.*, 171, 254-267, 2021.

3. **Ren Y.**, Y. Ding, Y. Zhang, and P. D. Christofides, "A Three-Layer Hierarchical Framework for Additive Manufacturing," *Digit. Chem. Eng.*, 1, 100001, 2021.

4. **Ren Y.**, Y. Zhang, Y. Ding, Y. Wang and P. D. Christofides, "Computational Fluid Dynamics-Based in-Situ Sensor Analytics of Direct Metal Laser Solidification Process Using Machine Learning" *Computers & Chemical Engineering*, 143, 107069, 2020.

5. Liu, T., C. S. Lough, H. Sehhat, **Y. M. Ren**, P. D. Christofides, E. C. Kinzel and M. C. Leu, ”In-Situ Thermographic Inspection for Local Powder Layer Thickness Estimation in Laser Powder Bed Fusion,” *Additive Manufacturing*, 55, 102873, 2022.

6. Wu, Z., A. Tran, **Y. M. Ren**, C. S. Barnes, S. Chen and P. D. Christofides, “Model Predictive Control of Phthalic Anhydride Synthesis in a Fixed-Bed Catalytic Reactor via Machine Learning Modeling”, *Chem. Eng. Res. & Des.*, 145, 173-183, 2019.

7. Ding, Y., Y. Zhang, **Y. M. Ren**, G. Orkoulas and P. D. Christofides, “Machine Learning-Based Modeling and Operation for ALD of SiO_2 Thin Films Using Data from a Multiscale CFD Simulation”, *Chem. Eng. Res. & Des.*, 151, 131-145, 2019.

Chapter 1

Introduction

1.1 Motivation

As the world enters the fourth industrial revolution, the incorporation of big data is becoming increasingly important because individuals constantly shift between reality and digital domains with the use of connective technology to enable and manage their work and daily lives [179]. As a result, data-intensive methods such as machine learning (ML) and deep learning (DL) rose in popularity due to their ability to assist and automate previously-considered difficult and tedious tasks. In engineering, ML and DL algorithms have been proposed in various industrial domains to handle tasks such as defect detection, data rectification, and process control [75, 134]. In this dissertation, the application of neural networks, a popular type of ML algorithm, within the additive manufacturing (AM) industry is explored with an emphasis on ML-assisted real-time process monitoring, data analytics, and data transfer.

Due to the thermal and mechanical complexity of AM processes, especially laser powder bed fusion (LPBF), process failures are often encountered, which may lead to defective parts or even fatal damage to the printing platform. For example, unexpected defocusing of the laser beam can cause cold and hot zones to appear on the powder bed. These heating abnormalities may induce thermal and mechanical stresses and eventually lead to the final build part to have poor mechanical properties, such as keyholing or lack of fusion [69]. The generated thermal stresses may also cause part distortion, which alters the powder layer thickness and ultimately interferes with the recoater arm on a layer-to-layer timescale [65]. Thus, it is important to identify these kinds of problems and control the process parameters to suppress them during the layer-to-layer printing process [82, 95, 109, 182, 184]. To monitor the printing process and to facilitate engineers to make crucial operating recipe updates, in-situ sensors are developed to investigate and record the printing process information in real-time [41, 83]. Some commercial examples of these sensors include the optical tomography (OT) and melt pool monitoring (MPM) sensors from the EOSTATE suite [148], the infrared (IR) sensors from FLIR Systems [100], and the physical powder bed images from P. Basler visible-light cameras [146].

However, there are several issues associated with sensor data that make them difficult to be utilized efficiently. The two major issues that need to be addressed are the overwhelming size of the data and the efficiency of data analytics to enable real-time process control. Therefore, a smart way of integrating sensor image analysis and efficient filtering of process data for transport and storage is needed. As demonstrated by [54], high-resolution camera sensors provide pictures at a very fast sampling rate, which produces terabytes of data per build. It is impossible to accommodate and store all information due to limitations in the speed of data transfer and storage space. In addition, it is also difficult for engineers to read and analyze every sensor image manually. To address the aforementioned concerns, end-to-end advanced data processing algorithms that connect the flow of process information from the initial data measurement to the final data storage are extensively researched. As a result, modern ML and DL algorithms can be implemented to integrate such functionalities as automated image processing and data transfer decisions. In particular, convolutional neural network (CNN) is one of the leading ML algorithm candidates for image analytics and can be applied to real-time monitoring of printing errors.

1.2 Background

Additive manufacturing was invented in the 1990s and has been exploited for the manufacturing of industrial parts using various kinds of materials including polylactic acid (PLA) plastic, photopolymer, gypsum, metal alloys, etc. [64]. In the past 20 years, the market share has grown by 20% each year, which created massive opportunities in the field of AM. In 2019, the AM industry reached approximately 10 billion dollars and was still evolving rapidly. Compared to conventional manufacturing techniques in the industry such as casting, AM technologies are significantly more advantageous due to their rapid prototyping time, fast manufacturing speed, reduced production cost, and versatile build geometry [120]. The adoption of AM techniques also demonstrates huge economic benefits, where a cost reduction of about 36% to 46% can be achieved [18]. Among the wide range of materials that can be manufactured with AM methods, metal alloys have particularly drawn attention in both academia and industry, and comprise 30% of the total AM market due to their use in high mechanical strength applications, such as tooling [40], pharmaceutical devices, and automotive equipment [90]. Metal additive manufacturing is estimated to have grown significantly each year, expanding at a compound annual growth rate of 27.8% and is projected to reach USD 5.51 billion by 2027. In particular, two major metal deposition methods are typically used in industry: directed energy deposition (DED) and laser powder bed fusion [60]. DED involves the melting of a powder or wire feed with an energy source to directly deposit a fused layer onto the substrate. It has the advantages of high printing speed and low-cost feedstock, but also faces low-resolution issues, causing significant part deformation and limited geometric capability. On the other hand, LPBF involves the use of a laser source to selectively melt the desired geometry on a bed of metal powder. Although LPBF is slower than DED, the precision and material strength of the parts made by the LPBF method is much superior to those made by DED. Therefore, LPBF becomes the more suitable option for applications with complicated design, stringent yield strength and high precision requirements, which are often encountered in the medical [150] and aerospace [99] industries.

In this work, a subcategory of LPBF called direct metal laser solidification (DMLS) is studied particularly. To construct the desired structure, the first step is to design the build geometry and the support structure, if necessary, with computer-aided design (CAD) software. Based on the 3D CAD geometry, a specific AM operating recipe is developed, including the layer-by-layer laser scanning pattern plan and the desired powder thickness for each layer. Before each layer is printed, the appropriate space in the deposition chamber is first created on the build platform by lowering the base of the platform by exactly one layer thickness. Next, the recoater blade pushes the metal powder from the fresh powder stock onto the building platform to prepare a uniform powder bed. Then, according to the predefined recipe, the laser power source scans across the layer to selectively melt the powder, and the molten powder subsequently re-solidifies to form the desired solid part. Despite the demand for high resolution, many part failures are encountered in industrial manufacturing due to errors and disturbances during the DMLS process. In particular, the thermal and mechanical stresses of the build part are highly dependent on the temperature, which is influenced by the laser power, the energy dissipation of the material phase change, and the conduction and convection within the deposition chamber. Thus, laser and cooling abnormalities may cause mechanical problems of the build part, such as lack of fusion or keyholing [69]. Furthermore, recoater streaking and jamming can occur due to undesired part deformation above the expected build surface [65]. Although it is desirable to understand and control the disturbances, the relationship between the variety of potential disturbances and operating parameters (e.g., scanning strategy, laser power, hatch spacing, and scanning speed) is very difficult to generalize for different materials and build geometries [43]. In addition to process parameters, the in-situ build properties of the process, particularly powder layer thickness, also have a significant influence on the local melt pool and overall properties of the part. In general, a thicker powder layer increases the production rate but can lead to porosity due to lack of fusion [121]. In addition, if the powder layer becomes too thin, it becomes discontinuous with vacancies that can negatively affect the porosity of the final part [30, 72, 149]. The thickness of the powder layer has been shown to affect the size of the melt pool [191], the grain size [103], the density [81], the microhardness [81, 103], the surface roughness [129], the strength, and the elasticity [154, 162] of the part produced.

To better understand the physics behind DMLS processes, several experimental works have utilized first-principles-based modeling and analysis methods for deterministic analyses. [16] utilized a multi-physics algorithm of heat transport and mechanical strength analytics to understand the influence of melt pool on the mechanical property of microstructure. [181] constructed an integrated model of energy density, material microstructure formation, and mechanical responses of the resulting parts. Also, other simulation models investigated the effect of operating parameters and strategies aided by FEM models. For example, [131] looked at different scanning strategies, and [49] investigated the influence of hatch spacing on the dimensions of the melt pool and the quality of the associated part. In addition, experimental studies endeavored to optimize the part design based on model-predicted properties. For example, [125] developed the approach of probabilistic rapid qualification design and utilized model-predicted results to reduce the overall qualification process time. [84] used the thermal profile predicted by the model and the stress analysis to formulate and optimize the design of the structural geometry and the support structure.

Nevertheless, existing simulation models are limited to batch-to-batch detection of defects and encounter difficulties in detecting defects during the real-time printing process. Thus, to avoid this problem, in-situ sensor monitoring technologies are developed to record the real-time manufacturing information. Many types of real-time monitoring technologies have been developed in academia and industry, including physical powder bed images, layer-wise temperature history maps, melt pool dimension data, etc. [39] developed an optical sensor setup that is connected with a field-programmable gate array (FPGA). This setup allows the transfer of sensor information at a sampling rate higher than 10 kHz. [57] provided a general review of valuable sensors that can be integrated with all types of metal additive manufacturing methods and applications. Moreover, the EOS company has developed advanced monitoring devices which consist of EOSTATE optical tomography, melt pool monitoring, base, and powderbed [148]. Most recently, extensive research has also been conducted on more efficiently utilizing these in-situ sensors through the integration of machine learning methods. [190] utilized support vector machine (SVM) techniques to extract plume and spatter information from powder bed images and evaluate their impact. [145]

used scale-invariant feature transform (SIFT) to extract features from the images and then used histogram of oriented gradients (HOG) to relate melt pool abnormality with the resulting defects. The same group also tried using CNN with transfer learning from an AlexNet backbone to classify physical disturbances like super-elevation and recoater jams. [185] exploited a semi-supervised learning algorithm, which bases the CNN model on the temporal ensemble method, to allow successful training with limited sensor images for bootstrap.

1.3 Dissertation Objectives and Structure

This dissertation presents the integration of machine learning into additive manufacturing, more specifically the DMLS and LPBF process, through applications in process monitoring/defect detection, data analytics, and data reduction and transfer. Specifically, the objectives of this dissertation are summarized as follows:

1. To develop high-fidelity DMLS FEM models to augment experimental data for training and maintenance of the CNN defect detection algorithm.
2. To develop automatic process monitoring and defect detection tools for DMLS using a cross-validated CNN algorithm trained from multiple sensor data.
3. To apply ML algorithm for the in-situ powder layer thickness estimation to prevent recoater abrasion.
4. To present a smart manufacturing framework focusing on the data transfer between machine, factory, and cloud for the additive manufacturing industry.

The dissertation is organized as follows.

Chapters 2 and 3 focus on the development of ML-assisted process monitoring tools for the DMLS process through the use of different sensors and FEM models. Chapter 2 discusses the construction of an FEM heat transfer model for DMLS and the utilization of the FEM model results to replicate different sensors. Specifically, the EOS OT and MPM sensor results are reconstructed and studied to detect different types of defects during the building process. To further automate this defect detection process, two CNNs are constructed to conduct real-time data analytics on each type of sensor images. Finally, the two CNNs are integrated together to develop a CNN cross-validation process monitoring algorithm. The cross-validation algorithm exploits the advantages of each sensor type and uses a meta-analysis method to make a final classification. A case study is conducted to demonstrate that the cross-validation algorithm outperforms an individual CNN trained from single sensor data.

The FEM model constructed in Chapter 2 is limited in application due to its reduced dimensions, which is caused by the large computational power associated with simulating realistic part proportions, and simplified geometry. Therefore, in Chapter 3, the modeling of the DMLS process is further developed through the construction of three multi-scale FEM models. At the highest level, a semi-arch with overhang is constructed with reference to experimental dimensions. However, using experimental-scale dimensions puts a heavy burden on the computational power necessary to run multiple FEM simulations. Therefore, micro- and meso-scale submodels are developed to describe and generalize the powder property and the laser behaviors, respectively. These generalized behaviors allow simplifications to be made within the experimental-scale model and thus lessen the computational burden. The experimental-scale FEM model output is then processed to replicate experimental LWIR sensor results. The FEM replicated sensor data is then used to augment experimental data for the development of a CNN process monitoring algorithm. This CNN process monitoring algorithm is further evaluated under experimental conditions with consideration of model accuracy and inference speed. Finally, the maintenance of the CNN algorithm is explored focusing on the transmission of new data to sustain an acceptable level of model accuracy.

Chapter 4 explores the application of ML for the detection of recoater abrasion through the in-situ estimation of local powder thickness. First, from first-principle equations, a 1D heat transfer model is developed to show the correlation between powder layer thickness and thermal properties.

This relationship is further validated by experimental data. Next, a neural network is applied for the prediction of powder layer thickness to improve performance. It is found that the neural network outperforms the experimentally derived relationship for low powder thickness regions where recoater interactions most often occur. Finally, the neural network-based powder thickness estimation method is applied to experiments where it displayed good performance for detecting recoater interactions due to uneven powder thickness.

Building on top of the ML-assisted process monitoring methods, Chapter 5 proposes a three-level framework regarding the data transfer between the machine, factory, and cloud for the additive manufacturing industry. At the machine level, on-line data collection and processing capabilities of individual machines are discussed. At the factory level, defect detection methods and predefined recipe update policies for machines are explored along with the coordination of data transfer between the two other connecting levels (machine and cloud). At the cloud level, the usage of cloud storage and research center for large recipe changes and development are presented. The proposed framework is further evaluated through a data transfer case study. Finally, the integration of the proposed framework with Industry 4.0 and its implications are discussed.

Chapter 6 presents an overview of the recent developments of time-series neural network modeling along with its use in model predictive control (MPC). A tutorial on the construction of a neural network-based model is provided and key practical implementation issues are discussed. A nonlinear process example is introduced to demonstrate the application of different neural network-based modeling approaches and to evaluate their performance in terms of closed-loop stability and prediction accuracy. Finally, the chapter concludes with a brief discussion of future research directions on neural network modeling and its integration with MPC.

Chapter 7 summarizes the main results of the dissertation.

Chapter 2

Computational Fluid Dynamics-Based In-Situ Sensor Analytics of Direct Metal Laser Solidification Process Using Machine Learning

In this chapter, we develop an automated data-flow that integrates the simulation model and a machine learning network for real-time process monitoring and sensor data analytics. The simulation models are tailored to incorporate possible disturbances, and sensor-specific outputs are generated from the simulation results to reflect realistic disturbances that might be identified in-situ during the manufacturing process. In particular, we first develop a model that simulates the heat transfer and phase change based on the operating parameter combination using COMSOL. Next, according to the EOSTATE OT and MPM measurements, the heat map sensor data are generated from the simulation model with disturbances introduced as classification categories. Then, two separate convolutional neural networks (CNN) are designed and trained with transfer learning with an AlexNet backbone to identify the problems that lie in the OT and melt pool images, respectively. In addition, the two CNN results are cross-validated using meta-analysis to make a

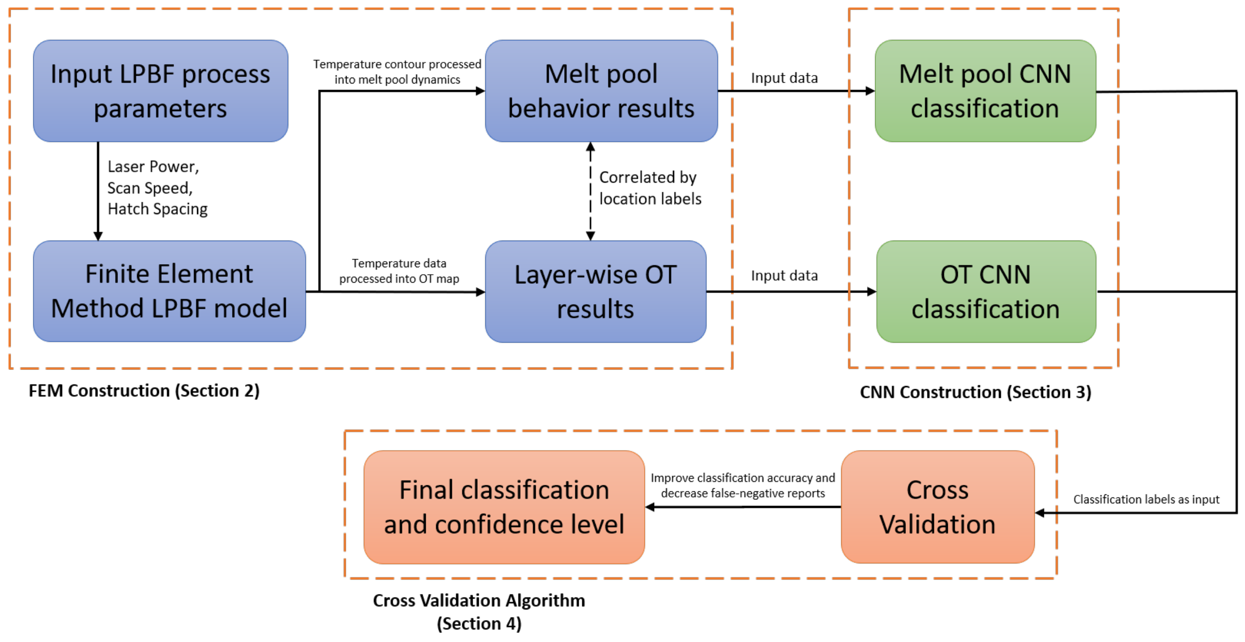

Input LPBF process parameters
Laser Power, Scan Speed, Hatch Spacing
Finite Element Method LPBF model
Temperature contour processed into melt pool dynamics
Temperature data processed into OT map
Melt pool behavior results
Correlated by location labels
Layer-wise OT results
FEM Construction (Section 2)
Input data
Input data
Melt pool CNN classification
OT CNN classification
CNN Construction (Section 3)
Classification labels as input
Cross Validation
Improve classification accuracy and decrease false-negative reports
Final classification and confidence level
Cross Validation Algorithm (Section 4)

Figure 2.1: Proposed automated data-flow which integrates FEM simulation and machine learning-based sensor data analytics.

final robust decision. The accuracy of the trained CNN and the computation time are demonstrated to be applicable to the in-situ AM process monitoring purposes.

2.1 FEM Model of DMLS and Thermal Sensor Output

As discussed in the introduction, first-principles-based numerical simulations have been utilized to model the DMLS processes to understand the effects of the operating parameters on the final build part properties. In the presented work, we specifically tailor our simulation model to predict and reproduce in-situ sensor monitoring information produced during the real-time manufacturing process. In this section, we first discuss the construction of a time-dependent FEM model that integrates heat transfer, the phase changes between powder, liquid and resolidified metal, the associated boundaries, and the operating recipes. Next, the performance of the developed model is validated with respect to experimental data and theoretical standards. Then, we will discuss the details of the melt pool and thermal OT sensors and the procedures taken to process the simulation results to reproduce realistic sensor outputs.

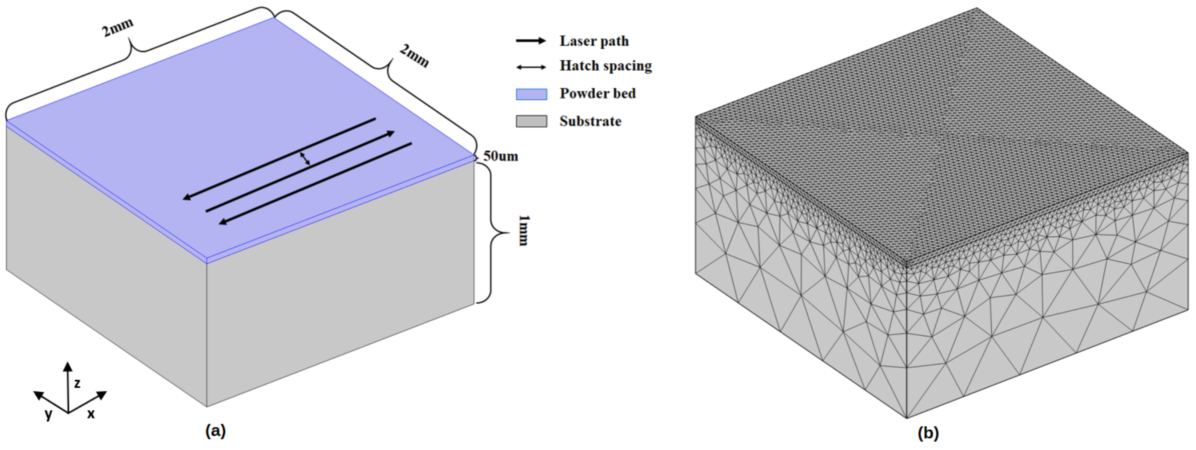


Figure 2.2: (a) Build part geometry and scan pattern. (b) FEM mesh scheme.

2.1.1 Build Part Geometry and Mesh

To ensure the generality of the developed method, a simple build geometry is adopted, which is a 1.4×1.5 mm rectangle with a height of 40 μm representing a single layer of IN-718 powder. The rectangular build part is printed on top of a square steel substrate with dimensions of $2.0 \times 2.0 \times 1.0$ mm in length times width times height. This is shown in Figure 2.2(a). The build part is positioned near the middle of the steel substrate so that there is sufficient room to the nearest boundaries to observe the heat transfer behavior.

As shown by the Figure 2.2(b), the FEM model geometry is first meshed with two different mesh resolutions. Due to the powder layer being the domain where most heat transfer takes place, we generate a finer mesh for this domain with free tetrahedral mesh elements of a maximum element size of 3.32×10^{-5} m and a minimum element size of 2.16×10^{-6} m. Since we are mostly interested in the *x*-*y* plane which is captured by the sensors, we employ a higher resolution factor for the *x* and *y* directions than the *z*-direction. For the steel substrate mesh, we use a much lower resolution mesh to enhance computational efficiency. The *z*-axis is dynamically meshed so that the mesh nearest to the powder bed is much smaller in size than the one at the bottom. Such an inflation method gives rise to a maximum element size of 4.76×10^{-4} m and a minimum element size of 1.01×10^{-4} m.

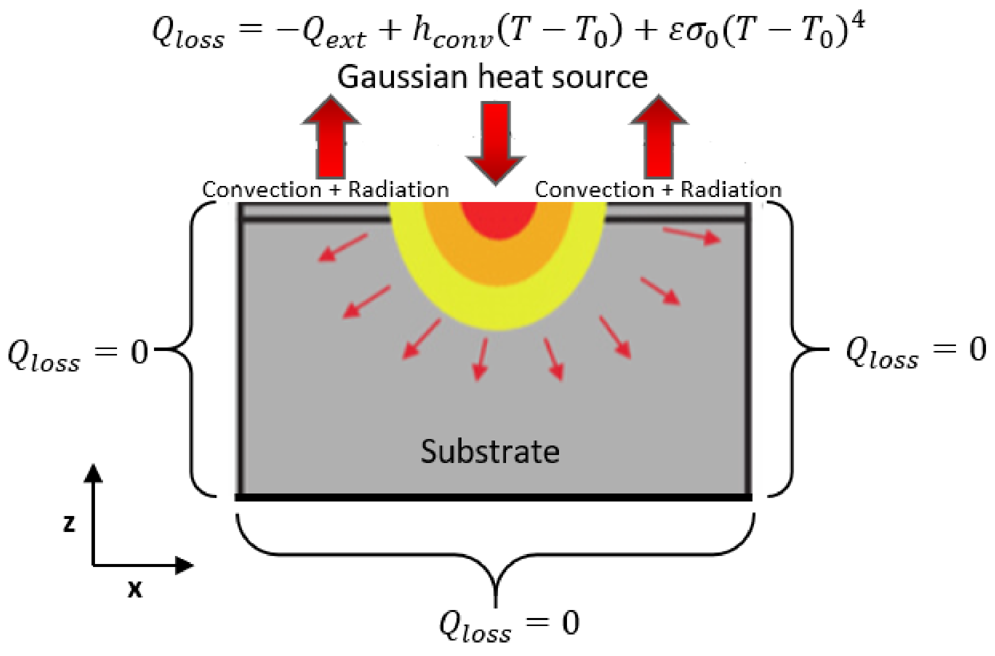


Figure 2.3: Boundary conditions applied to the build part geometry. Gaussian heat source, convection, and radiation is considered for the top surface while no heat loss is assumed for the remaining surfaces.

2.1.2 Thermal Energy Transport Governing Equations

In this work, we utilize the heat transfer module from COMSOL Multiphysics to capture the major thermal characteristics of the DMLS process. In the proposed model, we consider the heat transport (conduction, convection, and radiation) and its associated boundary conditions, the latent heat of phase change, and the temperature/phase thermal dependent material properties. It is important to note that the model does not consider the detailed microstructure formation and the possible fluid flow within the melt pool. Figure 2.3 summarizes the thermal boundary conditions used for FEM modeling in this study. The top powder surface considers the effect of the Gaussian laser source, convection, and radiation while the remaining surfaces assume no heat loss.

In each cell, the overall transient heat transfer is governed by the following heat balance equation:

$$\frac{\partial(\rho(T)C_p(T)T)}{\partial t} + \nabla q = q_s \tag{2.1}$$

where ρ is the density of the material, C_p is the specific heat capacity of the material, T is the temperature, and q_s is the rate of internal energy generation. Without the natural convection in the melt pool, the bulk energy transport is solely influenced by the conduction q as follows:

$$q = -\nabla(k(T)T) \tag{2.2}$$

where k is the thermal conductivity.

2.1.2.1 Boundary and Initial Conditions

The top surface boundary condition in the z-direction is described by the following equation:

$$-\frac{\partial(k(T)T)}{\partial z}(x,y,z=0) = -Q_{ext} + h_c(T-T_0) + \varepsilon\sigma_0(T^4 - T_0^4) \tag{2.3}$$

where h_c is the convective heat transfer coefficient, T_0 is the ambient gas temperature, ε is the emissivity, and σ_0 is the Stefan-Boltzmann constant. The heat loss due to convection and radiation is described by the second and the third term on the right hand side of the equation, respectively. In addition, the external heat source, Q_{ext}, is the laser source, which is assumed to have a Gaussian distribution:

$$Q_{ext}(r) = \frac{2AP}{\pi\omega^2} e^{-\frac{2r^2}{\omega^2}} \tag{2.4}$$

where A is the absorptivity, P is the laser power, ω is the laser beam radius, and r is the radial distance from the center of the laser beam.

For the other five surfaces, the assumption of no heat loss is assumed:

$$\frac{\partial(k(T)T)}{\partial x}(x=(0,l),y,z)=0, \quad \frac{\partial(k(T)T)}{\partial y}(x,y=(0,w),z)=0, \quad \frac{\partial(k(T)T)}{\partial z}(x,y,z=h)=0 \tag{2.5}$$

where l is the length of the build platform, w is the width of the build platform, and h is the height of the build platform.

In addition, the initial condition is that the temperature of the entire substrate at the beginning

$(t = 0)$ is at the ambient temperature of the printing chamber:

$$T(t = 0) = T_0 \tag{2.6}$$

2.1.2.2 Phase/Temperature-dependent thermal properties

In our model, we consider three different phases of IN-718 alloy including powder, re-solidified metal, and liquid, which all have distinctive thermal properties. For metal alloys used in AM process such as IN-718, the melting point is not at a single temperature due to their mixed composition. Instead, a transitional zone of solid or liquid exists, which is defined by the solidus temperature and the liquidus temperature. Any material below the solidus temperature is considered to be purely in the solid phase, and similar logic applies for the liquidus temperature. While the solid and liquid phase of the material can be differentiated through the inspection of temperature, the powder and re-solidified metal phase of the material both exist in the same solid phase temperature range. In this model, we attempt to differentiate between the powder and re-solidified metal phase using porosity, ϕ, which is a function of local temperature:

$$\phi(T) = \begin{cases} \phi_0 & \text{if } T \leq T_s \\ 0 & \text{if } T \geq T_l \\ \dfrac{\phi_0}{T_s - T_l}(T - T_l) & \text{if } T_s < T < T_l \end{cases} \tag{2.7}$$

where T_s is the solidus temperature, T_l is the liquidus temperature, and ϕ_0 is the initial porosity of the powder.

In addition, we are interested in the three temperature-dependent material properties that are important to the simulation: the specific heat capacity, the thermal conductivity, and the density of IN-718, whose profiles are shown in Figure 2.4. Figure 2.4 is plotted using experimental results reported from [137]. The heat capacity C_p is governed by functions of temperature at the solid and liquid phase, the latent heat of fusion L_f and the percentage at each phase θ also contribute to the

Table 2.1: IN-718 physical properties.

IN-718 properties	Value
solidus temperature	1533 K
liquidus Tempearture	1609 K
Latent heat of fusion	210 KJ/kg
Emissivity	0.35
Density, Specific heat, Thermal conductivity	See Figure 2.4

effective heat capacity as follows:

$$C_p(T) = \begin{cases} C_{p,s}(T) & \text{if } T \leq T_s \\ C_{p,l}(T) & \text{if } T \geq T_l \\ \dfrac{L_f}{T_l - T_s} + \theta_s C_{p,s}(T_s) + \theta_l C_{p,l}(T_l) & \text{if } T_s < T < T_l \end{cases} \tag{2.8}$$

where subscripts s and l refer to the solid and liquid phase, respectively. The key thermodynamic parameters are shown in Table 2.1. In contrast, the thermal conductivity and density of the powder and the re-solidified metal can be distinguished by porosity:

$$k_{powder} = k_{solid}(1 - \phi(T))^4 \tag{2.9}$$

$$\rho_{powder} = \rho_{solid}(1 - \phi(T)) \tag{2.10}$$

where k and ρ are the thermal conductivity and the density of the material, respectively. As stated in Equation 2.7, ϕ is dependent on temperature, which makes k and ρ also indirectly dependent on temperature.

2.1.3 Model Implementation and Verification

To validate the model, a range of combinations of the laser power and the scan speed were tested. While laser power and scan speed influence the resulting melt pool differently, they are often used in conjunction to calculate the total laser energy density which dominates the

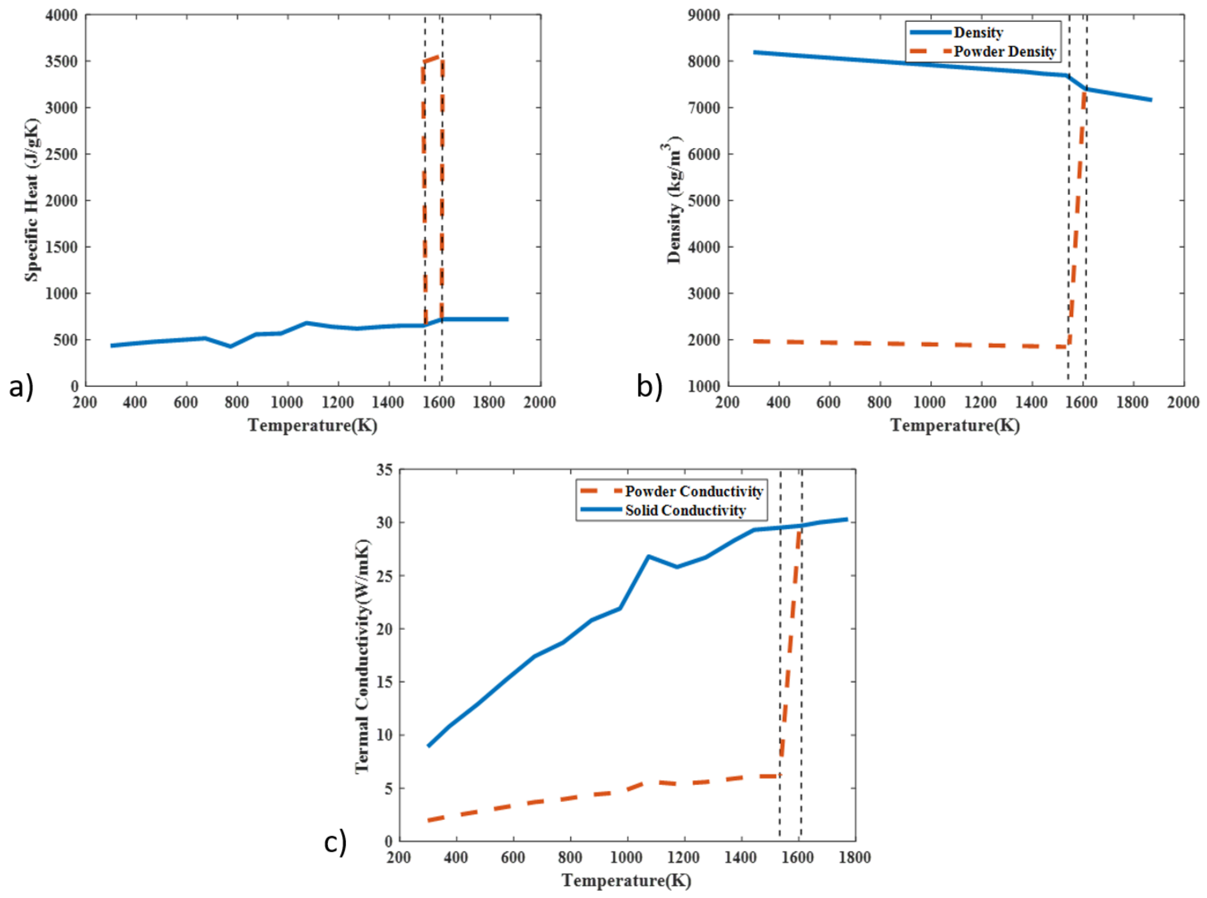


Figure 2.4: Temperature dependent thermal properties of IN718: (a) specific heat capacity, (b) density, and (c) thermal conductivity. The vertical dotted lines represent the transition region between the solidus and liquidus temperature.

Table 2.2: FEM model operating process parameters.

Process Parameters	Value
Laser power	70-300 W
Scanning speed	0.2-1.2 m/s
Hatch spacing	0.1 mm
Powder thickness	40 μm
Laser spot diameter	70 μm
Absorbance	0.6

comprehensive trend of the overall heating effect and melt pool dimensions. The energy density can be calculated as follows:

$$E = \frac{P}{V} \tag{2.11}$$

where E is the energy density, P is the laser power, and V is the scan speed.

Process parameters excluding laser power and scan speed are fixed at a constant 40 μm layer thickness and 0.1 mm hatch spacing. A commonly used bidirectional laser scanning scheme is chosen. In particular, the laser scans in the x-direction until reaching the build part's edge, jumps one hatch spacing in the y-direction, and then reverses the scanning direction along the negative x-direction. The range of operating conditions explored is listed in Table 2.2. The outputs collected from the FEM model are the melt pool dimensions and the temperature characteristics, which are compared to analytical solution to the Rosenthal equation and experimental findings.

2.1.3.1 Comparison with Experimental Results and solution to the Rosenthal Equation

First, we compare the melt pool dynamics developed in the FEM model with that of the experimental works [142], as shown in Table 2.3. It is demonstrated that the FEM model results closely resemble the experimental results for all operating combinations with an error smaller than 10%. One trend we observe is that the experimental values are constantly larger than the FEM results. The major reason behind this discrepancy between the FEM and experimental results can be attributed to the fact that the FEM model does not consider the influence of fluid flow of the molten metal in the melt pool. During the actual rapid heating process, the molten metal is driven

Table 2.3: Comparison between experimental and FEM model melt pool width.

Power (W)	Scan Speed (m/s)	Experimental (μm)	FEM (μm)
200	1.2	197.8 - 261.6	180.4 - 252.3
200	0.7	184.9 - 300.9	173.2 - 283.1
100	0.2	242.6 - 297.4	230.0 - 280.2

outwards due to the higher surface tension in the outer sections of the melt pool compared to the inner sections of the melt pool [102]. This phenomenon thus causes the experimental melt pool to be larger than the FEM model predicted melt pool.

In addition, the solution to the analytical Rosenthal equation is an important criterion that is often used to judge the melt pool dimension, which was originally developed for metal welding [138]. However, due to the similarity in the melting mechanism between welding and metal AM processes, the Rosenthal equation can be applied to describe the DMLS process. The following assumptions are made by the Rosenthal equation solution:

1. Thermophysical properties are temperature independent.
2. Quasi-stationary temperature distribution condition around melt pool.
3. Heat source is a point source.
4. Heat transfer is governed purely by conduction, ignoring convection and radiation.

Adopting these assumptions, the Rosenthal equation can be formulated as follows:

$$T = T_0 - \frac{\lambda P}{2\pi k r} exp\left(-\frac{V(r+\xi)}{2\alpha}\right) \tag{2.12}$$

where T_0 is the ambient temperature, λ is the absorptivity, P is the laser power, k is the thermal conductivity, V is the scanning speed, α is the thermal diffusivity, ξ is the x-direction moving coordinate expressed by $\xi = x - Vt$ as the laser moves along the x-axis, and r is the distance from the heat source defined as $\sqrt{\xi^2 + y^2 + z^2}$. The thermophysical properties of IN-718 for the above calculation are listed in Table 2.4.

Table 2.4: Ambient temperature (273 K) IN-718 thermal properties.

Thermal Property	Value
Absorptivity, λ	0.6
Density, ρ	$8220\ kg \cdot m^{-3}$
Specific heat, C_p	$43\ J(kg \cdot K)^{-1}$
Thermal conductivity, k	$11.4\ W(m \cdot K)^{-1}$

Table 2.5: Comparison between experimental and FEM model melt pool width.

Power (W)	Scan Speed (m/s)	Rosenthal Solution (μm)	FEM (μm)	Error
300	1.2	178.5	172.3	3.48%
200	0.7	190.8	173.2	9.23%
100	0.2	252.5	230.0	8.88%

Based on the Rosenthal equation, the width of the melt pool of low thermal diffusivity material including IN-718 can be estimated [157]:

$$W \approx \sqrt{\frac{8}{\pi e} \frac{\lambda P}{\rho C_p V (T_m - T_0)}} \tag{2.13}$$

where W is the approximate melt pool width, T_m is the melting point of the material, and ρ is the density listed in Table 2.5. The differences between the analytical solution to the Rosenthal equation and the FEM generated melt pool dynamics and their respective errors are shown in Table 2.3. In general, the Rosenthal equation predicts the melt pool width to be larger than the FEM melt pool width. An explanation for this discrepancy is the assumptions made by the Rosenthal equation. As shown earlier, the Rosenthal equation assumes the thermophysical properties to be temperature independent, and only conductive heat transfer is considered. Due to the lack of consideration of convection and radiation heat loss, the solution to the Rosenthal equation estimates a larger melt pool width than the FEM model.

2.1.4 Sensor Output Representation

After the thermal performances are validated, we use our proposed FEM model to reproduce the industrial monitoring sensor output: the melt pool monitoring (MPM) sensor and the optical tomography (OT) sensor. The MPM sensor reports the detailed behavior of the region where the laser focuses on at each timestep. In contrast, the OT sensor outputs monitor the overall temperature map of the whole layer. These two sensors can provide different views and information about the thermal features of the build platform. As shown in the previous section, the FEM model is capable of producing realistic thermal results at any time interval, which demonstrates a great potential to reproduce the realistic MPM and OT intensity map.

2.1.4.1 Melt Pool

The MPM sensor aims to capture the melt pool behavior through the implementation of an on-axis photodiode sensor or a short-wave infrared (SWIR) camera that follows the laser scan pattern. It is capable of capturing high-quality melt pool dynamics due to its on-axis configuration and small focal domain. A typical melt pool photodiode sensor has a very high capturing rate in the range of 10 kHz to 25 kHz. Due to this high capture rate, the MPM sensor can generally produce near-continuous process monitoring. However, the disadvantages of the MPM sensor include its limited field of view, the influence from scanning parameters, and the huge amount of data to be generated, stored, and analyzed. It has been demonstrated in our previous section that the FEM model can directly reproduce the melt pool with reasonable dimensions. Starting with this preliminary result, we can reproduce the result of the MPM sensor through calibrating our camera sampling rate to that of a real sensor by tuning the FEM model solution sampling rate to 1×10^{-4} s, i.e., 10 kHz. Thus, the model generates near-continuous melt pool images similar to those produced by a realistic sensor. An example of a single melt pool captured by the FEM model is shown in Figure 2.5.

2.1.4.2 Optical Tomography (OT)

The OT sensor also tries to capture the thermal aspect of the DMLS process but it aims to generate a layer-wise temperature map of each build layer. The OT camera can usually be an infrared (IR) camera and has several different types including long-wave infrared (LWIR) or

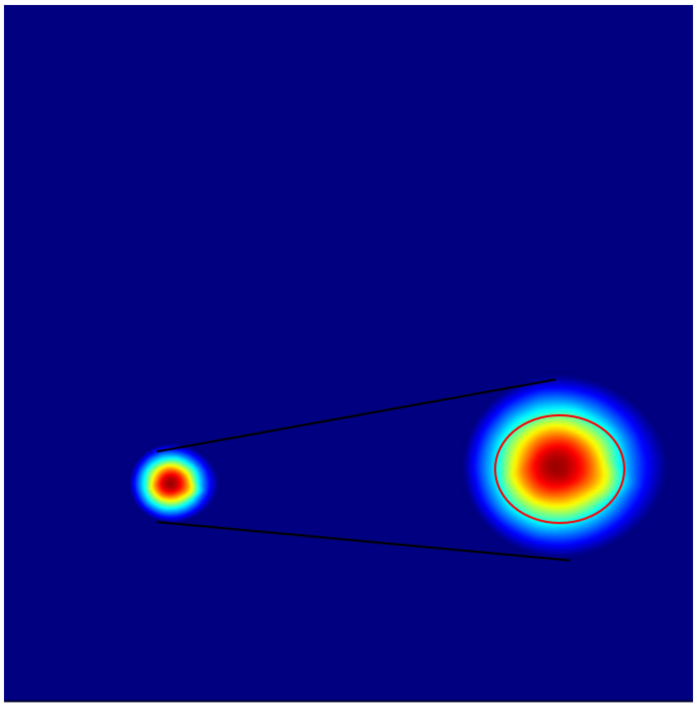


Figure 2.5: Example melt pool image at one timestep generated by FEM. The melt pool region is zoomed in and is highlighted by the red circle.

medium-wave infrared (MWIR). The OT sensor picks up the light intensity in a fixed sensor radius and processes these light signals into temperature values. Each temperature value is assigned to a region/pixel of the final image depending on the sensor specification. Under normal process operating conditions, the highest temperature values within the melt pool would contribute predominantly to the signal produced [117]. This is due to the nonlinearity of the temperature-intensity dependence relationship despite the rapid cooling of the material after the laser passes.

In order to reproduce the OT sensor results, we utilize the FEM model and the temperature information of individual mesh cells to create the layer-wise temperature map. Since the sensor radius is much larger than the individual FEM mesh cells, we average all of the values of mesh cells within the sensor radius and assign the averaged value as the final value for that sensor location at each capturing timestep. A sensor radius of 0.25 mm is used due to the build part's small dimensions. Then, we analyze the whole averaged temperature history of each previously defined radius and take the maximum temperature as the final value in our OT temperature map. An

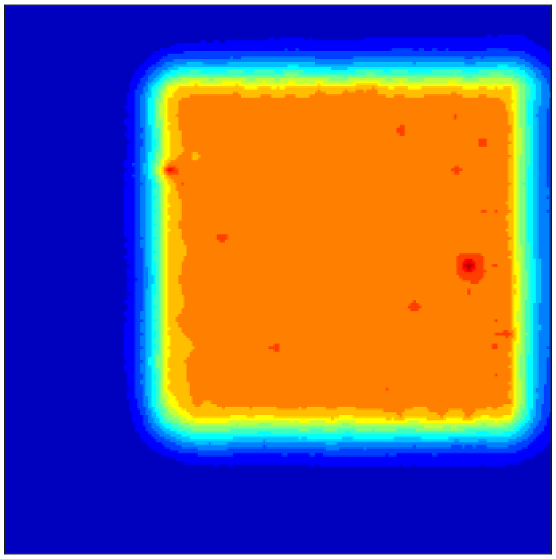


Figure 2.6: Single layer temperature mapping processed to reproduce OT sensor output. The red spots represent the potentially overheated regions in various sizes due to process disturbances.

example of a single layer OT temperature map is shown in Figure 2.6.

2.2 Sensor Data Analytics through Convolutional Neural Network

After melt pool and OT sensor images are collected, in-situ data analyses need to be constructed for real-time analytics, as discussed in the introduction. In this work, we adopt the convolutional neural network, a deep learning technique that is widely adopted for image-processing. Through training, CNN is able to automatically detect the disturbances and defects in the build part during the manufacturing process with high accuracy and efficiency. In this section, first, we will discuss the construction of CNNs. Next, the prediction results from the trained constructed CNN trained for both melt pool data and OT data are discussed.

2.2.1 CNN Construction

The major distinction of a convolutional neural network is the usage of convolutional layers to extract the correlational information from the high dimensional inputs, which makes CNN a good candidate for solving image recognition problems. A variety of trained general-purpose, high-performance CNN backbones, such as AlexNet, ResNet, and VggNet, have demonstrated to be able to produce highly-accurate image classifications. By changing only the final classification criteria and preserving the trained feature extraction functionality, CNN backbones can be easily modified and tailored to analyze DMLS sensor data. This process, which aims to specialize a pre-trained generic model, is known as transfer learning [67].

In this work, the AlexNet CNN backbone developed by [89] is used as the base CNN architecture, and transfer learning is performed using MATLAB for the purpose of in-situ sensor data classification. Figure 2.7 shows the data flow dimensions through the modified AlexNet CNN and Table 2.6 shows the specific dimensions of the proposed CNN. As shown in Figure 2.7, the hidden layers consist of five sets of convolution layers followed by pooling layers and fully connected layers. The ReLU function, i.e., $ReLu(x) = max(0,x)$, is applied between layers to introduce non-linearity, and Local Response Normalization (LRN) is performed to limit the unbounded output from the ReLu functions [89]. In addition, dropout layers are implemented to reduce network overfitting and computational resources needed by randomly deactivating neurons. The CNN terminates with a softmax layer followed by a classification layer. During transfer learning, only the dense layers are restructured and retrained to identify disturbances that are specific to the DMLS process using training data containing known errors in OT and MPM sensor data.

The exact structure and dimensions of the crucial layers are shown in Table 2.6. Specifically, the training data are the melt pool and the OT images generated from the FEM model and will be used to train their own respective CNN. The melt pool images are exported from the FEM model with a resolution of 227$\times$227 pixels in order to match the AlexNet input requirement. However, since each OT image represents the entire build platform, a resolution of 227$\times$227 pixels is too

Table 2.6: Modified AlexNet convolutional neural network with specific layer dimensions.

Layer	Dimension	Number of filters
Input	227×227×3	
Convolution	11×11×3	96
ReLU		
Channel normalization (LRN)		
Max Pooling		
Convolution	5×5×96	256
ReLU		
Channel normalization		
Max Pooling		
Convolution	3×3×256	384
ReLU		
Convolution	3×3×384	384
ReLU		
Convolution	3×3×384	256
ReLU		
Max Pooling		
Fully Connected		
ReLU		
Dropout		
Fully Connected		
ReLU		
Dropout		
Fully Connected	1×1×4096	
Softmax		
Classification	3	

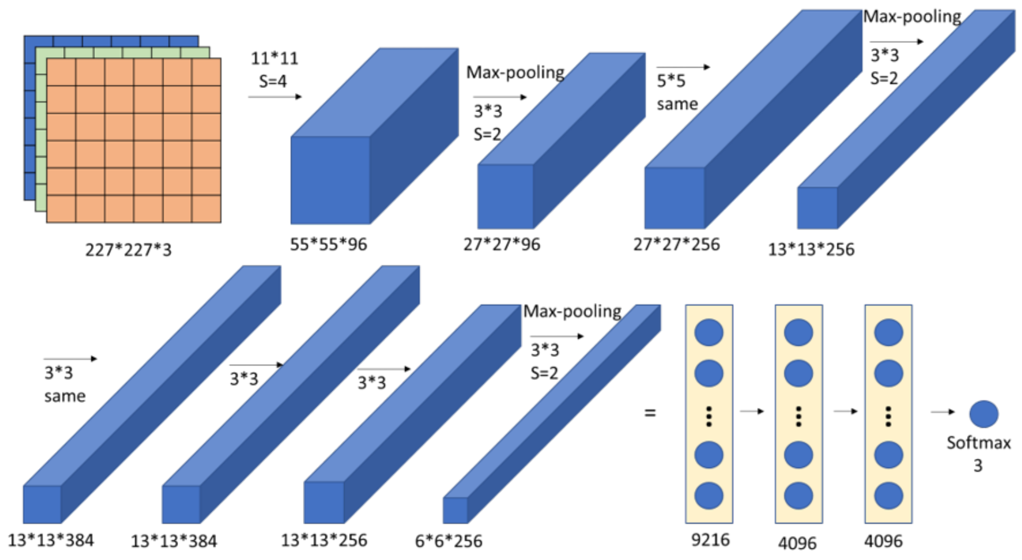


Figure 2.7: Modified AlexNet convolutional neural network structure.

low to capture all information. Therefore, through trials-and-errors, each OT image is dissected into 20×20 sub-regions of size 227×227 pixels to preserve all information in the original image, and each regional OT image is individually processed by the CNN.

After passing the initial input layer, the processed input images enter sets of combinations of convolution layers and pooling layers. By using trained weighted matrices known as filters or kernels, the convolution layers are able to recognize specific features. A CNN may contain multiple sets of convolution layers to extract different levels of features. It has been demonstrated that early filter layers recognize primitive features such as lines, edges, or corners, whereas latter layers can distinguish more abstract parts of the desired object, such as heads of animals and wheels of cars. This hierarchy of abstraction allows the implementation of transfer learning as features of low-level abstraction are often consistent throughout all geometries. For example, in the AM process, operational disturbances are usually distinguished by distinctive edges or lines in the heat map. Thus, a well-trained filter can be easily applied to various processes. A successful implementation of transfer learning may only require the re-adaptation of high-level filters or even just the final classification layers.

When a region of the input image is transformed by the filter, S_i, it is converted to a region on the

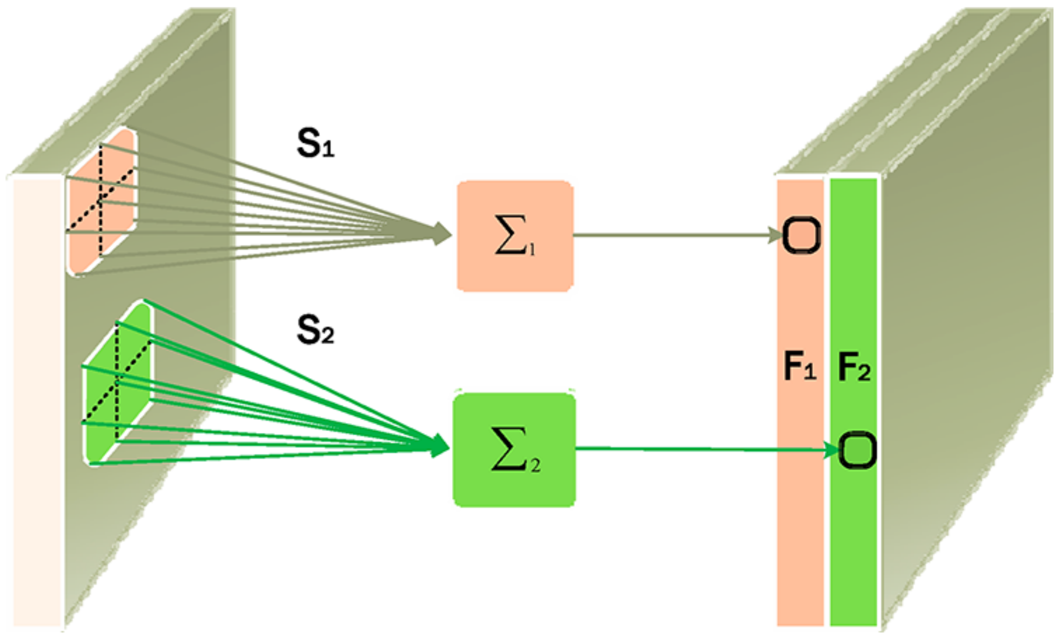


Figure 2.8: Convolutional process of transforming the input image using filters S to feature maps F.

output feature map, F_i, which retains the feature extracted by the certain filter as shown in Figure 2.8. Following the convolution layers, which expand the overall dimensions of the input data through the creation of multiple feature maps for each filter, pooling is performed to down-sample and summarize the feature map results [67]. In this work, the max-pooling down-sampling method is used, as shown in Figure 2.9. Max pooling extracts the maximum value from multiple pixels in a predefined region of the input layer, also known as the pooling window, and stores it in one pixel of the pooling layer. The process repeats as the pooling window slides across all of the input pixels, and the sliding distance is known as stride size. A 3×3 pooling window with a stride size of 2 is utilized in Alexnet.

After the convolutional layers and pooling layers have extracted features and reduced the overall dimensionality, fully connected layers are used to perform classification. Through training, each extracted feature from the convolutional layers is assigned weights with respect to each classification category. Then, the fully connected layers use the weight matrices to compute the confidence level of the input belonging to each category. In this work, the last fully connected layer has input dimensions of $1\times1\times4096$ pixels and it computes the confidence levels of the input

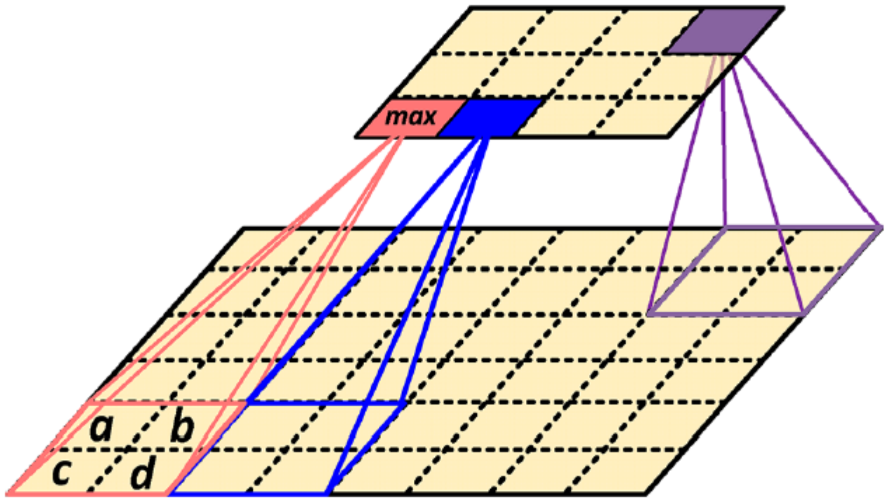


Figure 2.9: Max pooling process of an input convoluted layer.

images with respect to each of the three final categories. Finally, as shown in Figure 2.10, the output of the fully connected layer undergoes the softmax function, i.e., $\sigma(\vec{z}) = \frac{e^{z_i}}{\sum_{i=1}^{k} e^{z_i}}$ for $\vec{z} \in \mathbb{R}^k$, where the most likely category will be chosen as the final classification result.

Stochastic gradient descent algorithm is used to train the neural network model through the alternation of model weights based on the computed gradients. Specifically, the input training data is first forward-propagated to calculate the loss function:

$$L = -\sum_{i=1}^{3} y_i log(\hat{y}_i) \tag{2.14}$$

where L is the loss, y and $\hat{y}$ are the ground-truth label and predicted classification label, respectively. Given the loss function, the model weights are updated using the gradient computed from back-propagation [12]. Following this approach, the CNN is trained for multiple epochs until the desired convergence criterion is reached. One training epoch corresponds to the entire training dataset being propagated through once. The changes made to the weights during each training epoch are affected by the learning rate which determines how quickly the model adapts to the problem.

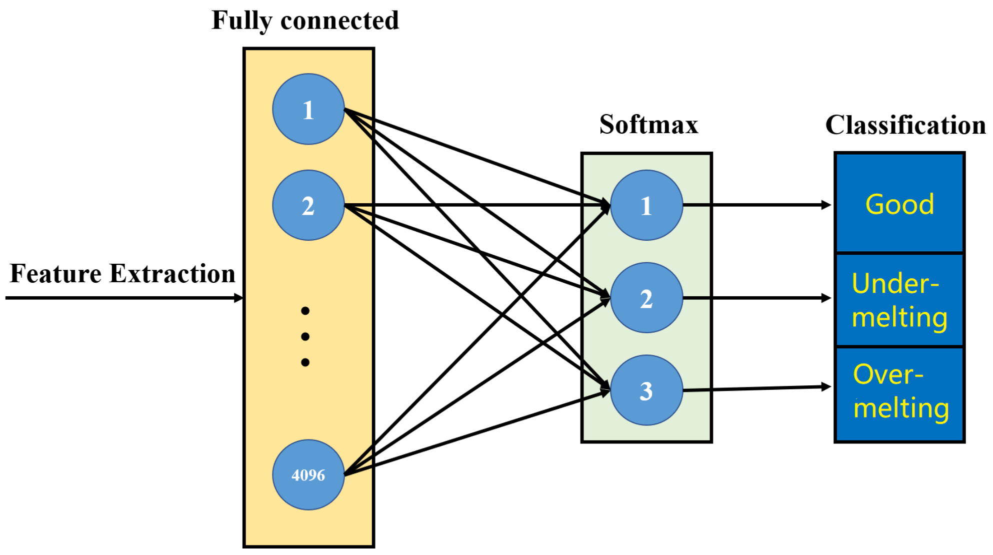


Figure 2.10: Fully connected layer with softmax and classification layers.

2.2.2 Individual Sensor Data Analytics through CNN

As mentioned in the previous section, both the melt pool and the OT CNNs share the same structure using the modified AlexNet backbone. The use of different sets of training images during transfer learning enables the melt pool CNN and OT CNN to recognize disturbances in melt pool images and OT images, respectively. In this section, we first discuss the selection details regarding each CNN's training and validation image sets. Next, we discuss the hardware used for training and the specific training options assigned for each CNN. Finally, we further test our CNNs and analyze the testing results of each CNN using the corresponding confusion matrix.

2.2.2.1 Melt Pool and OT CNN Training

Before either CNN is trained, training images must be labelled with ground truth, i.e., the categories that images actually belong to, and their respective locations on the build plate. Both sensor images are split into three categories: proper-melting, under-melting, and over-melting. A total of nine simulations are performed under various disturbances to collect melt pool and OT CNN training images. Disturbances are implemented by varying the laser power from 70 W to 437.5 W to create under-melting and over-melting regions. For the melt pool CNN, five simulation results are used for training where 1000 images are produced during each simulation. We notice that the melt pool image dataset contains much more proper-melting images than under-melting and over-melting images. Therefore, we carefully choose the proportion of images for each category and exclude excessive proper-melting images to ensure model accuracy by preventing class imbalance. The final melt pool CNN training dataset contains a total of 1412 different melt pool images, of which 710 correspond to proper-melting, 376 correspond to over-melting, and 326 correspond to under-melting. For OT CNN, all nine simulation results are used for training, where 400 regional OT images are produced during each simulation. Among these 400 regional OT images, 231 of them are images of the powder bed which the laser does not pass through and thus only 169 of them are images of interest. Similar to the melt pool CNN, regional OT images are selectively used for training to avoid class imbalance. The final training dataset for the OT CNN consists of 1173 regional images, in which 705 correspond to proper-melting, 274 correspond to over-melting, and 194 correspond to under-melting. Additionally, among the training image dataset for both CNNs, 70% of the images are randomly selected as training data while the rest 30% are used as validation data.

The OT CNN is trained for a total of 8 epochs with a learning rate of 2×10^{-4} as further training does not increase the CNN performance significantly. The total time to train the OT CNN is around 4 hours on a 6 GB Nvidia GeForce GTX 1060 GPU and the final training accuracy of the OT CNN is 86.2%. The melt pool CNN is trained for a total of 12 epochs. It is observed that the training of the melt pool CNN is more sensitive to the learning rate, which is likely to be due to the higher contrast in the melt pool images. Therefore, a lower learning rate of 1×10^{-5} is used. The total time to train the melt pool CNN is around 8 hours using the same GPU and the final training accuracy achieved is 89.3%.

2.2.2.2 Melt Pool and OT CNN Testing

In order to further test the accuracy and robustness of the trained CNNs, additional melt pool and OT images are collected and tested. The OT CNN test set consists of a total of 156 regional images, in which 86 correspond to proper-melting, 28 correspond to over-melting, and 42 correspond to under-melting. The melt pool CNN test set consists of a total of 678 melt pool images, in which 446 correspond to proper-melting, 135 correspond to over-melting, and 97 correspond to under-melting. The testing results of the two models are shown by the confusion matrices in Figure 2.11. The confusion matrix is commonly used as the metric to demonstrate the accuracy of classification results, which has the dimensions of $\mathbb{R}^{(N+1)\times(N+1)}$, where N represents the number of categories involved in the classification. The categories are laid out horizontally and vertically to represent the actual class and the predicted class, respectively. The first $N \times N$ entries of the confusion matrix show the counts of correct and incorrect classifications. The diagonal entries of the confusion matrix show the amount of correct classifications made by the CNN of each category. The remaining entries show the number of incorrect classifications and the specific types of errors made. The last row summarizes the accuracy given a specific ground-truth class while the last column summarizes the accuracy given a predicted class. Finally, the last (lower-right) entry of the confusion matrix summarizes the overall accuracy. According to the confusion matrix, the melt pool CNN is able to achieve an overall accuracy of 89.7% and the OT CNN achieves an overall accuracy of 84.0%. More specifically, for the OT CNN, 7.7% of the classifications correspond to false-positive reports, in which proper-melting images are classified as over-melting or under-melting. Additionally, 1.9% of the reports are false-negative reports, in which over-melting or under-melting images are classified as proper-melting, and 6.4% of the reports are misidentified errors of disturbances, in which over-melting images are labelled as under-melting or vice versa. For the melt pool CNN, there are no false-positive or misidentified errors for all errors are false-negative reports.

Although both CNNs yield acceptable overall accuracy, we observe that each CNN performs better when categorizing certain types of disturbances. This is due to the different focus on the disturbances by the MPM and OT sensors. The detailed breakdown of the OT and melt pool CNN performances is shown by the confusion matrices (a) and (b) in Figure 2.11, respectively. The OT CNN has an accuracy of 92.9% in categorizing under-melting cases while the melt pool

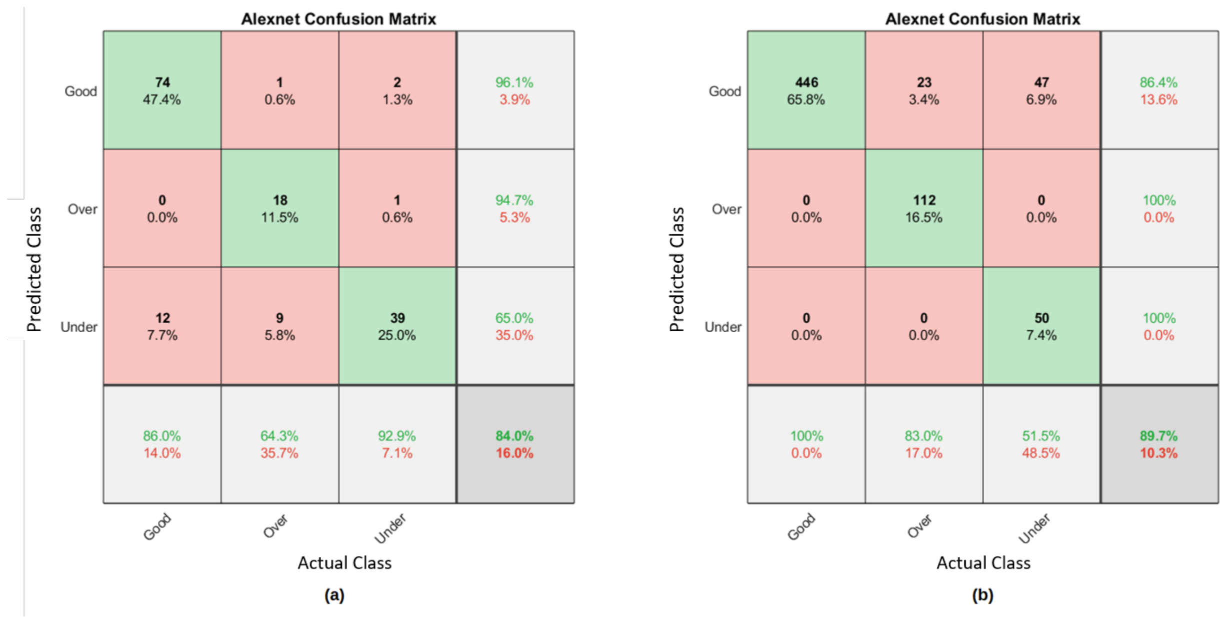


Figure 2.11: CNN testing confusion matrices: (a) OT CNN, and (b) melt pool CNN.

CNN has an accuracy of 100% and 83% in categorizing proper-melting and over-melting cases, respectively. Thus, it is desirable to combine the results of both CNNs to fully exploit their respective advantages. Therefore, a cross-validation scheme is formulated to further increase the overall accuracy.

2.3 Cross-Validation Utilizing Two Sensor Data Analytics

As mentioned in Section 2.1.4, the OT and MPM sensors have different fields of view. Also, it was demonstrated that the OT CNN performs better at identifying under-melting cases while the melt pool CNN performs better at identifying over-melting and proper-melting cases. Therefore, a cross-validation scheme is proposed to combine the strength of these two sensors and balance their CNN analytic bias. The framework and results of this cross-validation scheme are discussed in detail in this section.

2.3.1 Cross-Validation Scheme Formulation

In the cross-validation scheme, we attempt to classify regions of the build part using results from both the melt pool and the OT CNNs. Each region of interest is 0.1 mm $\times$ 0.1 mm in size, which is the same dimension as one regional OT image. First, both the melt pool and the OT CNNs will be used to separately classify each region and assign a confidence level to that classification. Then, if one region is classified differently between the two CNNs, the confidence levels from both CNNs will be used to cross-validate the classification results and determine a final classification. Additionally, a final equivalent confidence level is provided through meta-analysis.

The location labels are used to correlate the OT and melt pool images with the associated CNN labels. Since the melt pool images describe a smaller field of view than each region of interest, one region corresponds to six melt pool images. Therefore, all melt pool images corresponding to the same region are collected and compared to its ground truth. For each melt pool image, a confidence level is provided by the softmax layer, and a confidence level threshold is set to filter out melt pool images with low classification confidence levels. Based on the remaining qualified classifications, an empirically determined criterion is used to decide on the melt pool classification of a region, i.e., disturbances that span over three consecutive melt pool images can be correlated with disturbances on the regional OT images. Therefore, if there exist several potential classifications for a region, this empirical criterion is applied to determine the final classification. Additionally, the combined confidence level of the regional melt pool categorization is the average of the qualified categorizations, which is calculated as follows:

$$\mu = \frac{1}{n}\sum_{i}^{n} x_i \tag{2.15}$$

where μ is the average confidence, n is the number of qualified classifications, and x_i is the individual confidence level of each associated melt pool image.

After the melt pool CNN results are correlated with each separated region of the build part, we obtain the confidence level of the same region from the OT CNN and compare it with the results from the melt pool CNN. Since the dimensions of each region of interest are the same as the OT regional image's dimensions, the OT CNN classifications and confidence levels do not need to be further processed. Using the confidence levels from both CNNs, a cross-validated final classification is determined to be the higher classification confidence level result between the two CNNs. In addition, the final confidence level can be calculated through the use of inverse-variance weighting which is a meta-analysis method to combine the result of different studies on the same problem. It aims to minimize the variance through the calculation of the inverse-variance weighted average, $\hat{x}$, which is calculated as follows:

$$\sigma^2 = \frac{1}{n}\sum_{i}^{n}(x_i - \mu)^2 \tag{2.16}$$

$$\hat{x} = \frac{\sum_i^n(\frac{x_i}{\sigma^2})}{\sum_i^n(\frac{1}{\sigma^2})} \tag{2.17}$$

where $\hat{x}$ is the inverse-variance weighted confidence level and σ^2 is the variance of either the OT or the melt pool CNN.

2.3.2 Case Study

The proposed cross-validation scheme is validated with a case study using images from one layer of the DMLS process generated from the FEM simulation. A total of 1000 melt pool images and 169 OT regional images corresponding to that layer are cross-validated with each other. Individual CNN analysis is first processed and the classification results for the OT sensor analytics are shown in Figure 2.12(a). To prepare for the cross-validation scheme, the 1000 melt pool images are first labelled with their respective locations and correlated to each region of interest. These melt pool images are then categorized and given a confidence level using the previously trained melt pool CNN. Next, two confidence level thresholds of 72% and 87% are tested to eliminate unreliable classifications.

Figure 2.12(a) shows the detailed breakdown of the OT CNN classifying each region of interest.

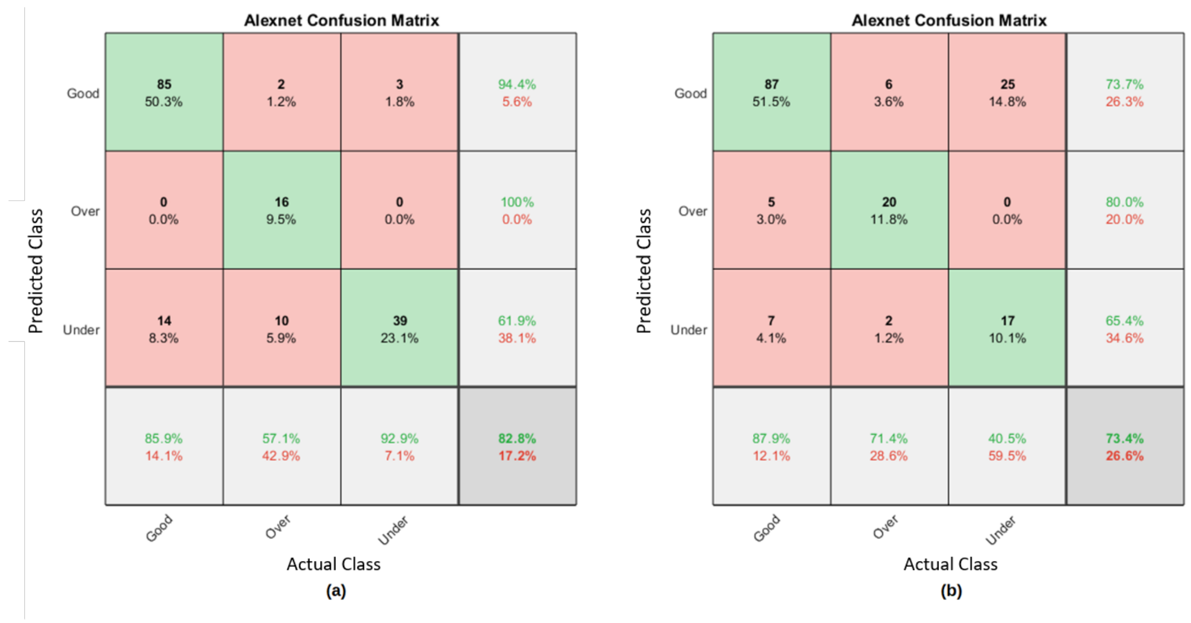


Figure 2.12: Confusion matrices of the (a) OT CNN and (b) clustered melt pool CNN classifying the 169 regions of interest.

The overall accuracy of the OT CNN is 82.8% which is similar to the previous testing accuracy. A total of 8.3% of the classifications are false-positive reports, 3% of the reports are false-negative reports, and 5.9% of the reports are misidentified errors of disturbances. Figure 2.12(b) displays the testing accuracy of the melt pool CNN categorization for each region of interest. The final accuracy is 73.4% with 7.1% false-positive reports, 18.4% false-negative reports, and 1.2% misidentified errors. While the melt pool CNN overall classifies the regions of interest less accurately than the OT CNN, it makes less misidentified errors than the OT CNN. Specifically, for the classification of over-melting images, the melt pool CNN has an accuracy of 71.4%, which is much higher than the 57.1% in the OT CNN. As a result, we apply a cross-validation decision scheme to improve the overall accuracy of the data analytics process.

Two confidence thresholds of 72% and 87% are tested for the cross-validation and their final decision accuracies are shown in Figure 2.13. According to the two cross-validated confusion matrices in Figure 2.13, the overall accuracy of the cross-validation is substantially higher than that of the individual, non-cross-validated CNN, shown in Figure 2.12. The number of false-positive, false-negative, and misidentified errors all have been reduced, which demonstrates

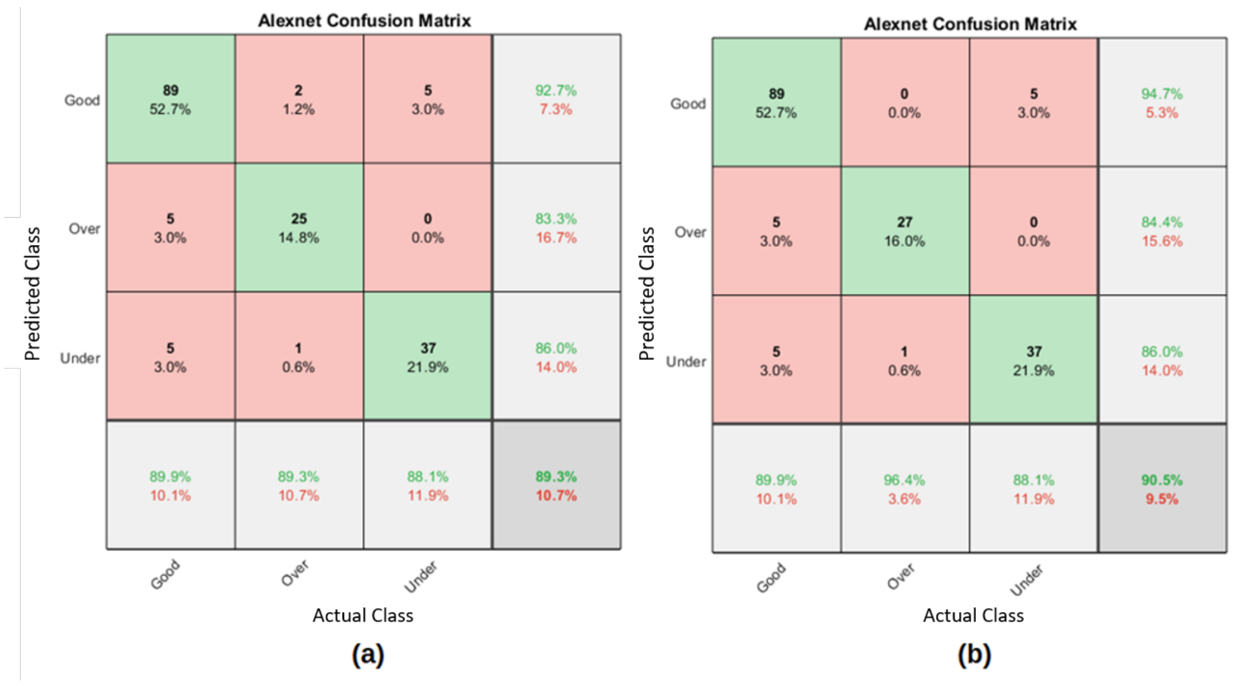


Figure 2.13: Confusion matrices showing the accuracy of the cross-validation scheme with confidence thresholds of 72% (a) and 87% (b).

that the cross-validation scheme successfully incorporates the advantages of both sensors. A comparison between all types of errors and the overall accuracies of the individual OT, melt pool and cross-validation methods is shown in Figure 2.14. The amount of misidentified errors drops from 5.9% when solely using the OT CNN to 0.6% when cross-validation is performed between the OT CNN and the melt pool CNN. The explanation for this decrease can be related back to the different type of sensor data used. As mentioned in Section 2.1.4, the MPM and the OT sensors have different fields of view and thus capture different aspects of disturbances. The cross-validation scheme utilizes information from both sensors and therefore is able to detect and classify disturbances more accurately. The amount of false-positive reports also decreases from 8.3% to 6% with the implementation of the cross-validation scheme. This decrease is not as significant as that of the misidentified errors due to the fact that both sensors face a similar performance issue when presented with false-positive errors. The major difference between the 72% and 87% cross-validation threshold scheme is the amount of false-negative reports. By setting a higher confidence threshold, the 87% threshold result is selected to contain more trustworthy

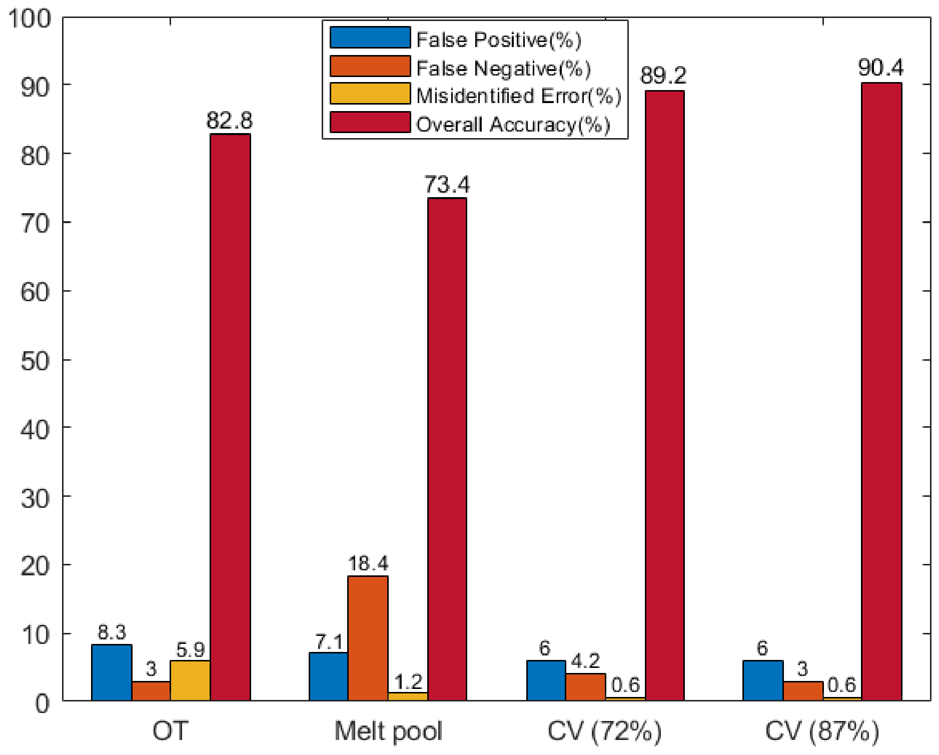


Figure 2.14: Bar graph showing the percent of false-positives, false-negatives, misidentified errors, and overall accuracy of all considered methods.

classifications than its 72% counterpart and thus resulting in a slight decrease in false-negative reports. At the same time, the 87% threshold results also contain less data than the 72% threshold result as labels with confidence level between 72% and 87% are discarded. This may result in the cross-validation scheme being less robust due to the limited data pool size as more results are discarded when the confidence level threshold is increased.

2.4 Conclusion

In this work, an integrated cross-validation framework using CNNs for the in-situ processing of the DMLS process was constructed. First, the DMLS process was simulated using COMSOL Multiphysics, a physics-based simulation software, to obtain the melt pool behavior and layer-wise OT image data through finite element method modeling. Specifically, the FEM model described the heat transfer behavior of the DMLS process, accounting for phase change, conduction, convection, and radiation. Then, the FEM model results were validated against both experimental data and analytical solutions. Next, the thermal data from the validated FEM model were processed into two sets of images corresponding to the output format of the two types of realistic sensors: the melt pool sensor and the optical tomography sensor. Specifically, the temperature contour map at each timestep reproduced the output from the MPM sensor and the locally averaged temperature history of the build layer reproduced the output of the OT sensor. Afterwards, these two sets of images were labelled and used to train their respective machine learning model for each sensor. A convolutional neural network was chosen to be the main machine learning technique used in this work. Transfer learning was performed on the AlexNet CNN backbone for each sensor, and both trained CNNs achieve reasonable testing accuracy individually. A single-layer case study demonstrated that the classification accuracy of identifying OT images using OT CNN alone was 82.8%. To further improve the classification accuracy, a cross-validation scheme was formulated and applied to the model sensor outputs. Based on the statistical analysis of the classification results of the melt pool and the OT CNNs, the cross-validation scheme determined a final label as well as a final confidence level. With the implementation of the cross-validation scheme, the testing accuracy increased up to 90.4% while the amount of misidentified errors decreased by 5.3%. It can be concluded that this cross-validation approach of utilizing CNN for in-situ DMLS process monitoring shows promise as it accurately predicts disturbances and can be potentially applied to real-time manufacturing monitoring platforms.

Chapter 3

Finite Element Modeling of Direct Metal Laser Solidification Process: Sensor Data Replication and Use In Defect Detection and Data Reduction via Machine Learning

In this chapter, a part-scale finite element method (FEM) model is developed to investigate the heat transfer behavior of the DMLS process and to extract experimentally relevant thermal features. Specifically, a microscopic and a mesoscale sub-model are initially developed to describe powder properties and the laser behaviors, respectively, and their outputs are directly incorporated in the part-scale FEM model. The FEM model-generated data are then processed to replicate the long-wave infrared camera (LWIR) sensor outputs. Finally, both the experimental and FEM model-generated images are used to train the machine learning algorithm for in-situ defect detection applications. In addition, we are endeavoring to look into a convolutional neural network utilizing such realistic thermal features from LWIR images, with and without a variety of simulated data augmentation to examine its contribution to the CNN training. The thermal feature extraction and the machine learning algorithm are then utilized for the data reduction purpose. Also, the transmission strategy is demonstrated to filter out a significant amount of redundant data while maintaining high model prediction quality.

3.1 FEM Modeling for DMLS

As introduced in the previous section, there has not been an existing part-scale simulation model for the DMLS process. The major blocking factors include the high computational demand and the long computational time. Thus, we first review the validity of a previously developed mesoscale heat transfer model in [135]. Then, to reduce the computational demand of the large-scale model, complexity reduction strategies are investigated with microscopic models, including a Gaussian heat source reduction to an equivalent uniform heat source, and a radiation simplification using a comprehensive 1-D transport model. Next, we associate these two model outputs to our heat transport modeling framework for experimental build part investigation and thermal feature analysis. In addition, parametric analyses for various operating parameters, geometric variations, and disturbances are carried out using cluster computation. The heat transfer module from COMSOL Multiphysics is used to capture the major thermal characteristics of the DMLS process.

3.1.1 Mesoscale Finite-element Method (FEM) Model Setup

To investigate the heat transport on the mesoscale, a detailed model has been constructed with a small-sized simple build geometry in [135], as shown in Figure 3.1(a). Thermal boundary condition assumptions are considered for the mesoscale FEM modeling. In particular, the top surface consists of a powder layer, where the laser is applied. A Gaussian laser source is considered, as well as convection of the ambient gas and the radiation. In contrast, no heat loss is assumed for the remaining surfaces, the substrate, and the peripheral sides, shown in Figure 3.1(b). The time-dependent heat transport is dictated by the following equation with the assumption of no

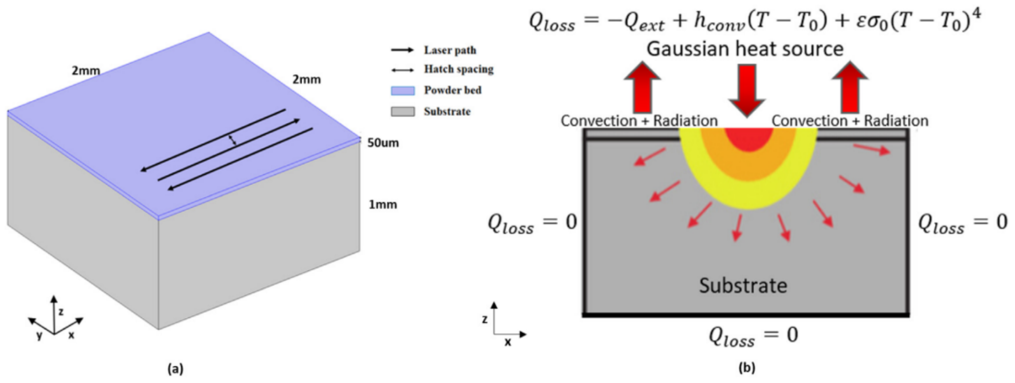


Figure 3.1: Mesoscale build part geometry: (a) 3D view and scan pattern, (b) Cross-sectional view with boundary conditions.

convection in the melt pool:

$$\frac{\partial(\rho(T)C_p(T)T)}{\partial t} - \nabla(k(T)T) = 0 \tag{3.1}$$

where ρ is the density of the material, where k is the thermal conductivity, C_p is the specific heat capacity, and T is the temperature. The top boundary condition consists of convection, radiation, and laser heating:

$$-\frac{\partial(k(T)T)}{\partial z}(x,y,z=0) = -\frac{2AP}{\pi\omega^2}e^{-\frac{2r^2}{\omega^2}} + h(T - T_b) + \varepsilon\sigma_0(T^4 - T_b^4) \tag{3.2}$$

where P is the laser power, A is the absorptivity, ω is the laser beam radius, and r is the radial distance, h is the heat transfer coefficient, T_b is the ambient gas temperature, ε is the emissivity, and σ_0 is the Stefan's constant. For the other surfaces, no heat loss is assumed, as shown in the following equation:

$$\frac{\partial(k(T)T)}{\partial x}(x=(0,l),y,z) = 0, \quad \frac{\partial(k(T)T)}{\partial y}(x,y=(0,w),z) = 0, \quad \frac{\partial(k(T)T)}{\partial z}(x,y,z=h) = 0 \tag{3.3}$$

where w, l, h are the width, length, and height of the build platform.

Three different phases of alloy are considered which include powder, liquid, and re-solidified metal. The powder properties change irreversibly once the powder has been melted. A transitional zone of solid or liquid exists between solidus temperature, T_s and liquidus temperature, T_l. The thermal conductivity k, specific heat capacity C_p, and the density ρ of the material, are governed by the latent heat of fusion L_f and the ratio of each phase θ:

$$C_p(T) = \frac{L_f}{T_l - T_s} + \theta_s C_{p,s}(T_s) + \theta_l C_{p,l}(T_l) \text{ if } T_s < T < T_l \tag{3.4}$$

where subscripts s and l refer to the solid and liquid phases, respectively. In addition, the powder phase thermal properties k_{powder} is defined in terms of the porosity, ϕ:

$$k_{powder} = k_{solid}(1 - \phi(T))^4 \tag{3.5}$$

which is dependent on temperature:

$$\phi(T) = \begin{cases} \phi_0 & \text{if } T \leq T_s \\ 0 & \text{if } T \geq T_l \\ \dfrac{\phi_0}{T_s - T_l}(T - T_l) & \text{if } T_s < T < T_l \end{cases} \tag{3.6}$$

The key thermodynamic parameters and their functional forms are shown in [135] and in Figure 3.2 [137]. The resulting melt pool dynamics are compared with that of the experimental works from [142]. It is demonstrated in [135] that the two results deviate less than 10%. Also, the model confirms the analytical estimate from the Rosenthal equation [138], which is investigated in detail in [135].

3.1.2 Microscale Investigation for Computational Demand Reduction

Despite the accuracy provided by the mesoscale simulation, it is computationally heavy to directly incorporate the developed model into a larger-scale FEM model. The two aspects that

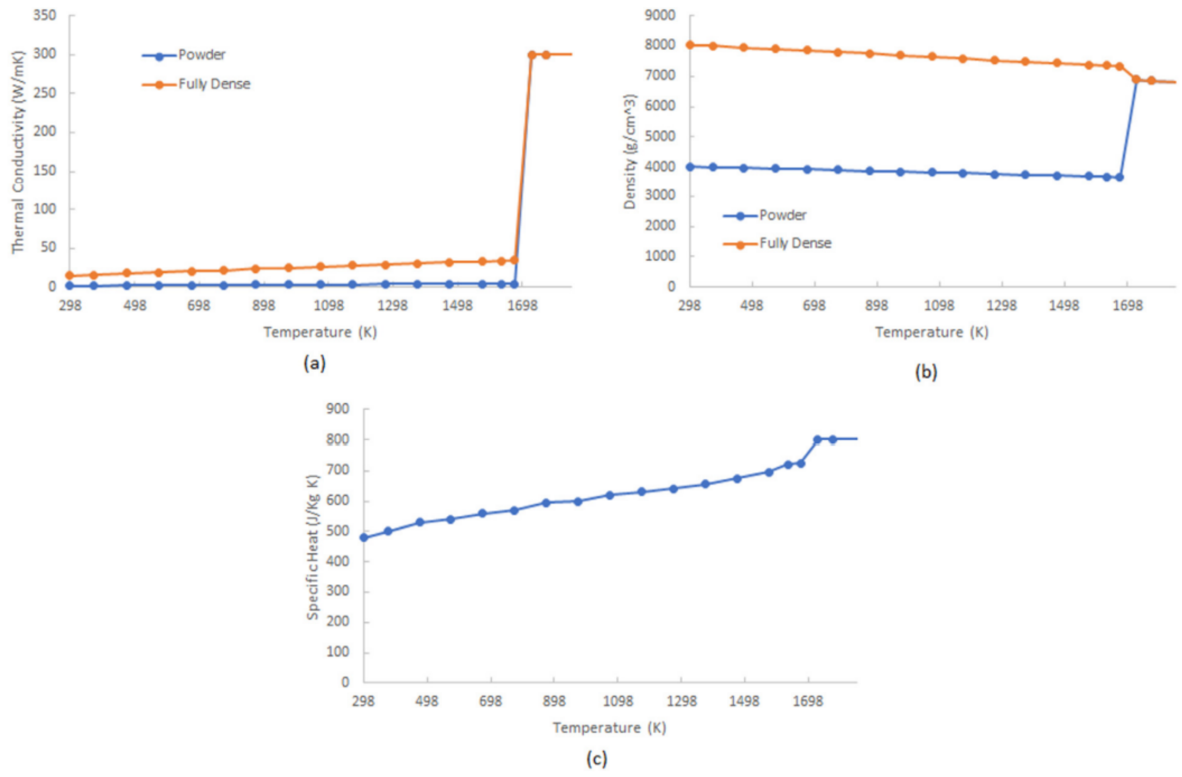


Figure 3.2: Temperature dependent thermal properties of Stainless Steel 304L: (a) thermal conductivity, (b) density, and (c) specific heat capacity.

Table 3.1: FEM model operating process parameters.

Process Parameters	Value
Laser power	200 W
Exposure time	75 μs
Point distance	65 μm
Hatch spacing	0.1 mm
Powder thickness	40 μm
Laser spot diameter	70 μm
Absorbance	0.7

especially require large quantity and high-quality meshes are the Gaussian laser source and the radiation. Thus, we are looking for valid simplification to efficiently characterize these two phenomena without losing model fidelity.

When looking at a model that incorporates the entire build part, the focus range is around 0.25 - 0.65 *mm* in radius, as prescribed by the pixel dimension of the LWIR camera, which will be explained in further detail in Section 3.1.3.1. Thus, the exact heat map around the laser focus does not need to be precisely reproduced as in the Gaussian type of laser simulation. Instead, a uniform heat source is considered, which assumes a constant heat flux inside the beam radius, and zero heat flux outside, as shown in the following equation:

$$q(x,y,t) = \begin{cases} \frac{2P}{\pi R^2} & \text{if } r \leq R \\ 0 & \text{if } r > R \end{cases} \tag{3.7}$$

where q is the heat flux simulating the uniform laser distribution, P is the laser power, R is the laser diameter, r is the location of the laser in relation to the x and y coordinates, the velocity of the laser movement, v, and time, t. A temperature comparison is made between the two kinds of laser sources under the same power. The maximum temperature difference is less than 3% for a moving laser, and more importantly, a pixel-wise average temperature difference is less than 4%. The reciprocal of step size is around 5×10^4 with and 8×10^4 without this simplification.

Thus, it can be concluded that there is no significant change in temperature and little change when producing sensor images by using the assumption of a uniform laser distribution. Under such uniform laser assumption, the mesh size demand is significantly reduced and the number of mesh points can be cut by about 80%.

In addition, to evaluate the effect of radiation and to seek a solution to efficiently describe its contribution, we apply a 1-D heat transfer model for the IN-718 powder/air and Stainless Steel 304L powder/air systems as shown in Figure 3.3. A closest packed structure is considered for the powders with uniform size according to experimental and industrial relevant recipes. Constant temperature boundary conditions are applied at the two sides, which are around the laser

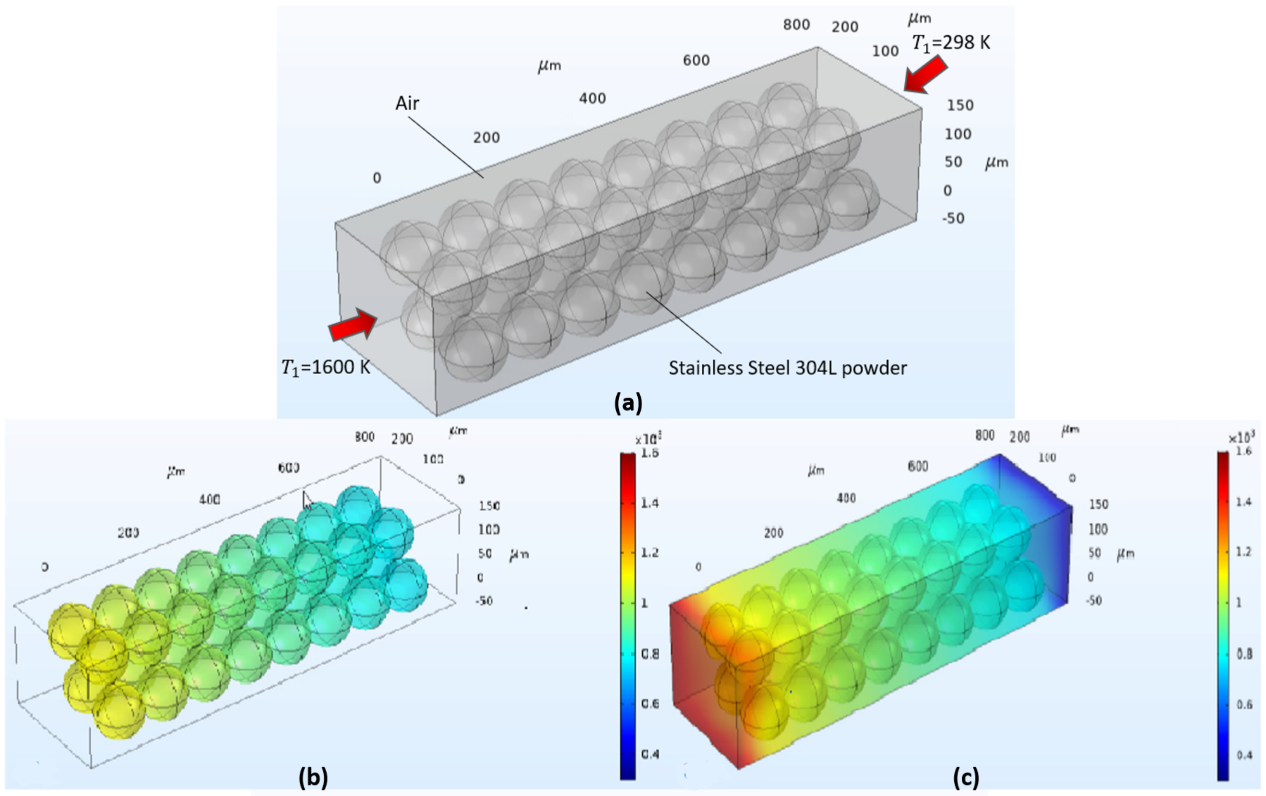


Figure 3.3: (a) The air and closely packed powder system. Temperature distribution: (b) Only in the powder, and (c) both in powder and in air.

temperature and the ambient temperature, respectively. Conduction is applied in both powder phases and air, whereas natural convection is considered in the air. Radiation is considered at the front and on every powder beads surface. It can be seen from Figure 3.3 that the temperature gradient decreases sharply before the first contacting surfaces of powder due to radiation, and the air is the primary source of conductive heat transfer after the first powder surface. Thus, an effective thermal conductivity of powder k_{eff} can be considered for the porosity, conductivity of air and powder, as well as for the effect of radiation. It has been demonstrated that the calculated value lies within the prediction from [119].

3.1.3 Experimental Based Part-Scale Thermal Feature Analysis

After the thermal performances are validated, we apply the developed FEM scheme with respect to experimental data from MST to reproduce results from the longwave infrared (LWIR) thermal sensor, which monitors the overall temperature map of the whole layer. As shown in the previous section, the mesoscale FEM model is capable of producing realistic thermal results at any time interval. In addition, aided by the microscale and mesoscale model outputs, a significant level of computational demand can be reduced so that part-scale geometry can be simulated. In addition, in order to ensure model convergence, a dynamic mesh is implemented, which simultaneously allows for mesh Independence and an efficient yet accurate solution. These modifications demonstrate a great potential to reproduce the realistic thermal feature map and sensor output. Thus, we adapt our FEM model to that of the experimental plan of a desired 15 *mm* by 20 *mm* by 10 *mm* semi-arch geometry, as shown in Figure 3.4. As presented in Table 3.1, a pulsed laser is simulated according to the experimental setup, which is governed by the point distance, the movement between each laser pulse, and the exposure time, the duration between each laser pulse. Other numerical specifications and details are similar to [135].

3.1.3.1 Experimental Temperature Feature Analysis

As shown in Figure 3.4, the target geometry is a semi-arch with varying degrees of overhang. The experimental plans and trials were conducted at Missouri University of Science and Technology (MST) and we aim to create FEM simulations that produce similar results as the experimental trials. We are particularly interested in recreating the defects relating to the thermal aspects of the build process. This can be a disturbance in the laser causing a region of over-melting or under-melting, or a geometry-based defect caused by its large overhanging angle. The heat transfer rate between materials is mainly governed by the thermal conductivity, k, of the material. During the DMLS process, the thermal conductivity of the powder k_{powder} and the solid, k_{solid}, differ around one or two orders of magnitude resulting in the solid being much better at conducting heat than its powder counterpart. At the bulk region, the heat from the laser quickly dissipates through the solid metal beneath it. However, at overhanging regions, the heat from the laser cannot be dissipated in time due to being surrounded by mainly powder material, causing a heat saturation problem. As a result of this heat oversaturation, over-melting may occur at this region, when the

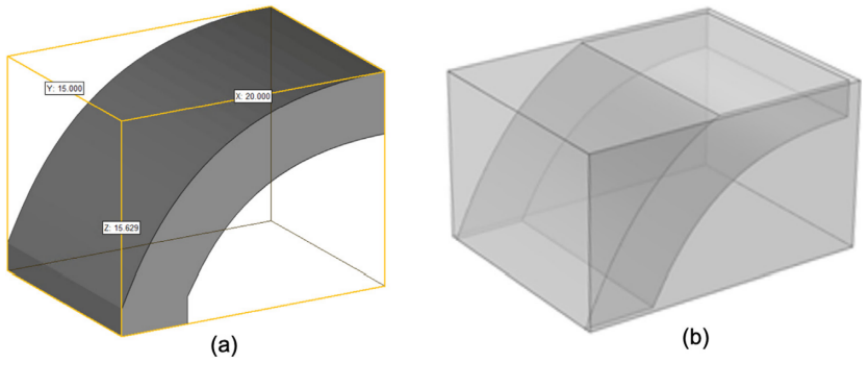


Figure 3.4: (a) Experimental semi-arch geometry, and (b) FEM simulation model geometry.

laser repetitively scans near this region before heat is fully dissipated. This in turn can cause a bad surface finish or even collapse of the part. In order to detect sample quality relating to the thermal aspects of the DMLS process, an LWIR camera (FLIR T420) is installed above Renishaw AM 250 vaccum-tight build chamber with a ZnSe window to monitor the defects caused by the overhang.

The LWIR camera attempts to generate a layer-wise 2D thermal history of each build layer. It picks up the radiation in a fixed field of view (0.65 mm/pixel) and calibrates these radioactive signals into temperature values. Each temperature value is assigned to a region/pixel of the final image depending on the sensor specification. Under normal process operating conditions, the highest temperature values within the melt pool would contribute predominantly to the signal produced [117]. This is due to the nonlinearity of the temperature-intensity dependence relationship despite the rapid cooling of the material after the laser passes. The LWIR camera has a frame rate of 30 Hz which in turn generates thousands of images for one layer of the build process. The maximum temperature of each pixel can be extracted from thousands of images and remapped into a single layer-wise image. This image can then be easily stored and analyzed in real-time by machine learning methods to detect defects. Also, this maximum temperature feature allows us to spot any non-uniform temperature distribution along the surface of the build part. A heat saturation region caused by large overhanging angles can be directly reflected as shown in Figure 3.5.

3.1.3.2 FEM Data Processing and Validation

To reproduce the LWIR camera results produced by the experiments, we need to further process our FEM model-generated data. After extracting the time-series temperature data from the FEM model, we calibrated the timestep of the FEM model to that of the actual LWIR camera capture rate. The FEM model generates temperature values at a rate of 300 Hz which is ten times faster than the capture rate of the experimental LWIR of 30 Hz. Therefore, the FEM raw data is temporally averaged by taking the mean of every ten data points to reflect this difference in capture rate. After the FEM data is in the same temporal frequency as the experimental data, we also need to map the FEM data to the same spatial domain as the pixel dimension. The last step is similar to the maximum temperature feature extraction step of the experimental data processing step. While the spatial averaging step is performed for every time step, in this work, we are focusing on the maximum temperature achieved by each pixel space. Therefore, for the final surface pixel temperature mapping, only the maximum temperature of the entire printing period is saved. The data calibration and preprocessing step is done within MATLAB similar to the experimental work.

We evaluate the FEM model-generated sensor image both qualitatively and quantitatively with respect to the experimental sensor images. For qualitative comparison, we look for general trends in the temperature distribution displayed in sensor images. For quantitative comparison, we examine the difference in temperature values for each pixel. It should be noted that qualitative comparison is more important due to its machine learning applications. This is because we do not want duplicate images to supplement our experimental training set, instead, we want images that share the same general trends but with some noise. In the experimental pictures shown in Figure 3.5(a), the heat saturation effect causes a zone of high temperature on the right edge corresponding to the overhanging region of the build part. A similar trend of high temperature on the right edge of the FEM generated image is reproduced on the FEM simulation, as shown in Figure 3.5(b).

To compare the sensor images quantitatively, we take one set of images from layer 212 of the build part and examine the temperatures by each pixel. The maximum temperature region represents the temperature of the overhanging zones while the minimum temperature region

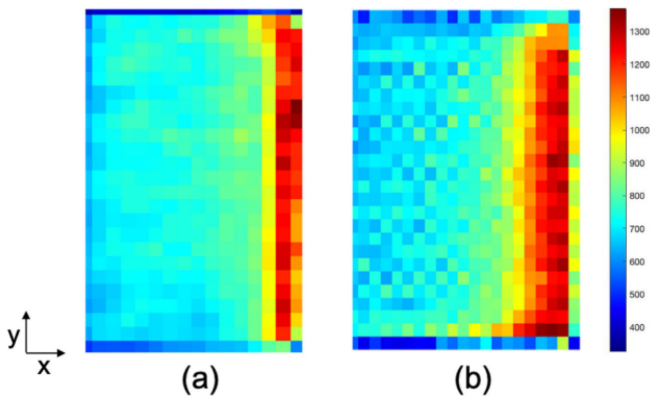


Figure 3.5: Top view of (a) experimental sensor image, and (b) FEM generated sensor image.

represents the bulk part above printed metal. It can is demonstrated that the maximum temperature difference is around 9% while the minimum temperature difference is around 8.2%. This difference is within a reasonable range as the experimental images' temperature can also vary due to different camera calibrations and non-uniform correction methods. We are more interested in the qualitative comparison between the images and if the temperature trends make physical sense.

3.1.4 Parametric Analysis using Cluster Computation

The aforementioned part-scale FEM model serves as a general-purpose guideline for DMLS modeling, and it is readily customizable with respect to model accuracy and computational demands. In this work, we are also trying to look beyond experimental analysis with the aid of simulation. In particular, defects and disturbances are costly to be investigated in experiments, as those errors lead not only to higher material and operating costs but also detrimental damage to the printing platform. The purpose of the CNN classifier is to detect defects in real-time and having more types of defects in our training dataset will improve its performance. Therefore, we want to leverage the part-scale FEM model to intentionally create disturbances in order to augment the existing training dataset. Even though the FEM model can generate data at a lower cost than experimental builds, it is also not optimal to arbitrarily generate a massive dataset solely for data augmentation. The FEM model should aim to generate rare or machine damaging defect data that AM machines cannot produce in a large capacity. The first type of defect implemented is the excessive overhang which is caused by the build geometry. This type of defect can cause oversaturation of heat near the overhang due to the powder's low conductivity of heat. This can lead to a bad surface finish and even collapse of the part. The second type of defect we implemented is line streaking which is caused by scanning pattern and laser power surge. We created this defect in our FEM model by applying a laser power increase for three to five hatch spacings. This can lead to surface protrusion which can potentially cause recoater arm jams and damage. The final type of defect we implemented is the corner saturation defect. This type of defect is generally caused by suboptimal scanning patterns and laser movement errors. We created this defect in the FEM model through the slow turning of the laser at corners along with a suboptimal scanning pattern. Similar to line streaking, corner saturation can cause surface protrusion which can lead to recoater jams. Figure 3.6 shows a summary of the defects introduced in the FEM model along with visualizations.

Due to a large amount of potential operating conditions, geometries, and disturbances scenarios to explore, a variety of simulations need to be carried out. To realize such mass simulation production, parametric analysis from COMSOL is utilized. Furthermore, to enhance the simulation computational efficiency, job distribution is realized with distributed cloud computing, where more than 100 of 8 GB RAMs are allocated on the UCLA hoffman2 cluster. The number of jobs and adopted cores is based on the mesh memory demand, the available core on the cluster, and the parallel computing overhead.

3.2 Machine Learning using CNN

Pretrained CNN backbones are used for transfer learning to investigate the DMLS process adopting the simulated and experimental thermal features. In order to show the importance of using FEM model simulated images, we will be training two different types of CNNs, one with simulation augmentation and one without, to compare their respective performance in classifying the testing datasets.

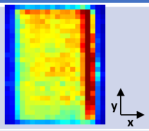
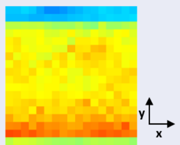
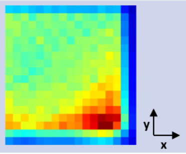
Type of Defect	Augmentation Methodology	Cause	Top-view result	Effect
Excessive overhang	Created by using the same geometry as experimental. No external disturbances added.	Caused by too much overhang in geometry. Most common type of defects included in simulation.		Bad surface finish. Extreme cases may lead to part collapse.
Line streaking	Created by applying 3-5 hatch spacing of laser power increase. Causes a line of over-melting at certain locations.	Caused by outside disturbance and potentially scanning pattern.		Bad surface finish. Extreme cases can lead to extrusion which lead to recoater jams.
Corner saturation	Created by slow turning (pause of 5-8 exposure time) of laser at corners of part. Causes heat saturation at corners.	Caused by scanning pattern.		Bad surface finish. Extreme cases can lead to extrusion which lead to recoater jams

Figure 3.6: Table showing the three types of defects introduced in our FEM simulation.

3.2.1 Dataset Preparation

Our total dataset contains images from four different sources. One is the FEM model simulated semi-arch images, and the rest are the experimental images of cantilever beam and semi-arch images from two different builds. All images first undergo data processing before they are separated into the training and testing datasets. For our training dataset, we use one set of experimental semi-arch images, and the FEM model simulated semi-arch images as augmentation. For our testing dataset, we use the other set of experimental semi-arch images and the experimental cantilever beam images. In our testing dataset, we wish to evaluate how well our CNN classifier performs on classifying known geometry from a different build and an unknown geometry. As a result, three different training datasets are created, composed of different combinations of experimental and simulated images. The three different sets of data are all used to train CNN classifiers in order to display the effectiveness of the simulated data in helping the detection of defects. Set 1 contains only experimental data and is used as a baseline to compare with if there were not any simulated data to use as augmentation. Set 2 and set 3 both contain half and half of each type but differ in their simulated data composition. Set 2 only contains geometry-based defects such as those caused by overhangs while set 3 also contains intentionally created defects such as lines and edges caused by laser power disturbances. It is important to note that the FEM model generated dataset should not be used as a replacement but as a supplement to the experimental dataset. In this work, the ratio between the simulated and experimental data is kept below 0.5, so that the training dataset distribution is more realistic and closer to the experimental builds than the FEM simulations.

In any AM build, without any major machine and build design problem, there is little room for error for the majority of the layers. As a result, defects only start appearing toward the latter layers, often causing the build to stop. This trend is reflected in our experimental dataset, which causes there to be much more non-defective data than defective data. When training any machine learning model, we want to avoid the problem of class imbalance as much as possible as it can lead to the classifier having an inherent balance toward one class. Thus, a sampling strategy is included when selecting our training dataset in which we oversample the less saturated class (defective images) and undersample the more saturated class (non-defective images). Other ways to combat the class imbalance would be to adjust the weights when learning one class to the other while training. However, in our work, we found that using a sampling method was enough to solve the class imbalance problem and achieving good performance.

3.2.2 CNN Construction and Training

A residual network (ResNet) backbone is used as the basis for the transfer learning of the CNN classifier. Residual networks contain many residual blocks which create skip connections between layers, which allow the CNN to classify both simple and complex features without overfitting. In a typical residual network, there are two different types of skip connections: the identity block and the convolutional block, shown in Figure 3.7(a) and (b) respectively. In the identity block, there exists a shortcut path from the input that allows the skipping of three convolutional blocks which consists of a 2D convolutional layer, a batch normalization layer, and sometimes a Rectified Linear Unit (ReLU). Similar to the identity block, the convolutional block allows the input to skip over three convolutional blocks but the shortcut consists of a single convolutional block. Both of these

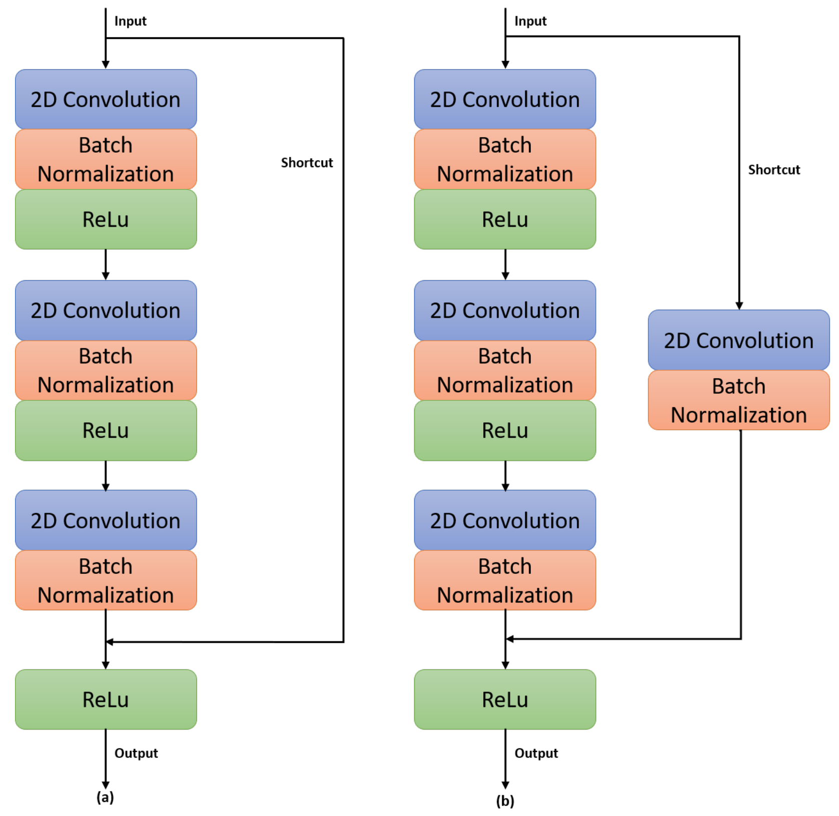


Figure 3.7: Two types of skip connections in residual networks. (a) The identity block skip connection, and (b) The convolutional block skip connection.

skip connection types allow the input to skip many computations depending on the classification problem. Due to the implementation of these skip connections, residual networks are able to be constructed much deeper than other neural networks without adding extra computation power and worry about parameter overfitting. Among the popular CNN structures, we found the ResNet-18, a variation of the residual network, architecture to work best due to its ability to train deeper networks while also maintaining good performance.

After selecting the backbone architecture for our CNN classifier, we applied transfer learning on the last three classification layers to make it specialize in classifying LWIR sensor images. Next, we conducted hyperparameter tuning (learning trade, maximum training epochs, and the type of optimizer) on a subset of the training images in order to save computational cost. The convolutional and pooling layer hyperparameters are not changed, such as dimensions, strides, and channels, as during our hyperparameter tuning, we found the original combinations performed better. Instead of the traditional grid search algorithm, we utilized a random "coarse-to-fine" search algorithm as it allows for more efficiency by not training CNN with suboptimal hyperparameter values multiple times.

3.2.3 CNN Model Evaluation

After our CNN classifiers are trained from the three different training sets, we compare the difference in their performance on the pre-constructed testing dataset. All three CNN classifiers achieve similar high training accuracy, around 96%, which indicates that they can accurately distinguish between the defective and non-defective classes with the same distribution as the training set. Therefore, the main purpose of the testing dataset is to evaluate how well our CNN classifiers handle less known geometries and defects.

The detailed performance of each CNN classifier is shown in Figure 3.8, which demonstrates a clear difference in the testing accuracy achieved by the three different training sets. The CNN trained with only experimental data (Set 1) has the lowest testing accuracy which likely indicates overfitting. Specifically, it achieves accuracy below 50% in classifying different builds and new geometries which indicates that it fails to generalize beyond one specific build. Set 2 contains FEM simulated images to provide a bit more noise to reduce overfitting. The CNN trained from set 2 achieves around the same training accuracy as set 1 while having a higher testing accuracy of 75.4% from 46.4%. Upon a close inspection of the error set 2 has made, we realized that set 2 has trouble with classifying defects coming from different sources than the training. For example, the CNN trained from set 2 has difficulties in detecting the corner and edge defects from the cantilever beam images. Within set 3, the common intentionally created defects are added to the training dataset. As a result, the CNN trained from set 3 achieves the highest testing accuracy of 94.5% while also maintaining a good training accuracy.

A common method in comparing different classifiers is through the plotting of the receiving operating characteristic (ROC) curve. The ROC curve plots the true positive rate against the false positive rate at different classification thresholds. The final ROC curve for the augmented

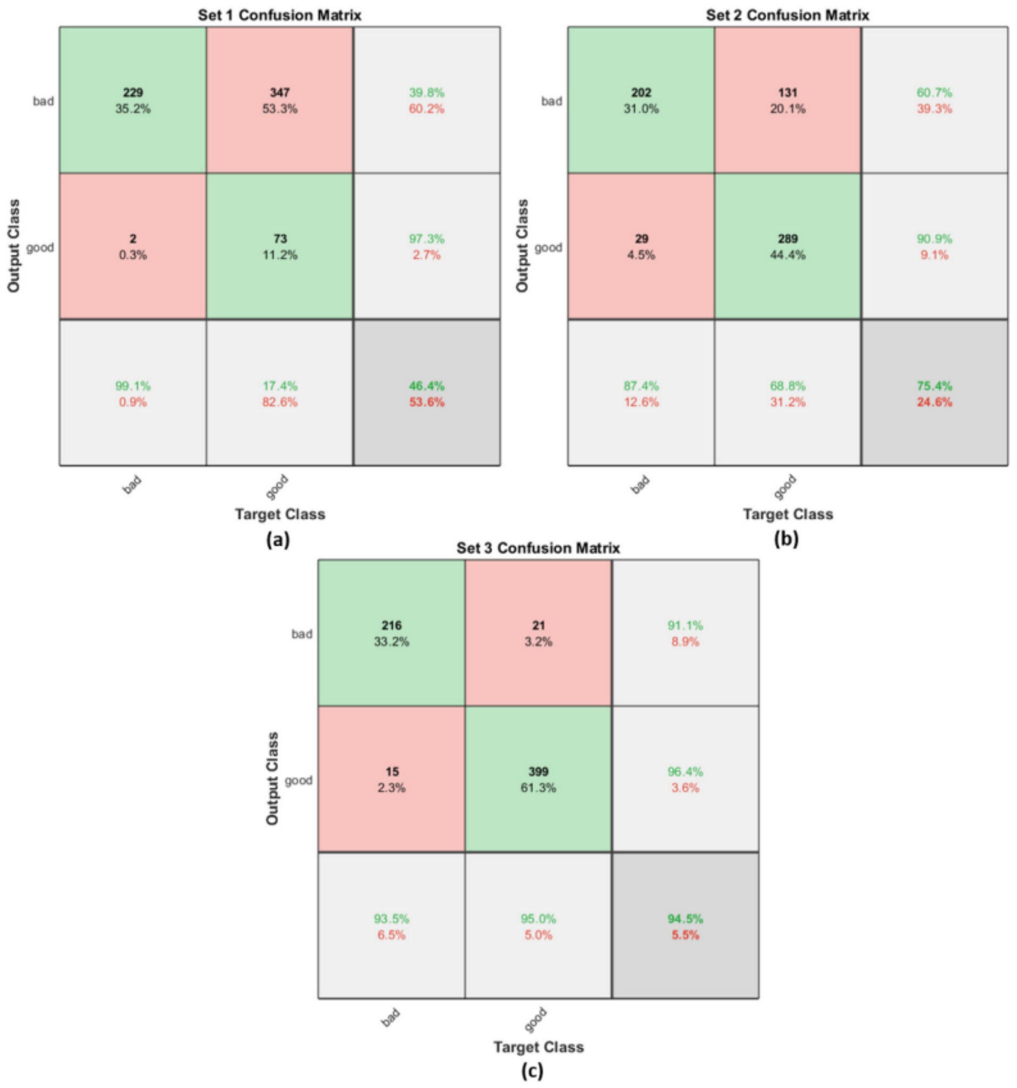


Figure 3.8: Confusion matrices for the CNN classifiers trained by (a) set 1, (b) set 2, and (c) set 3.

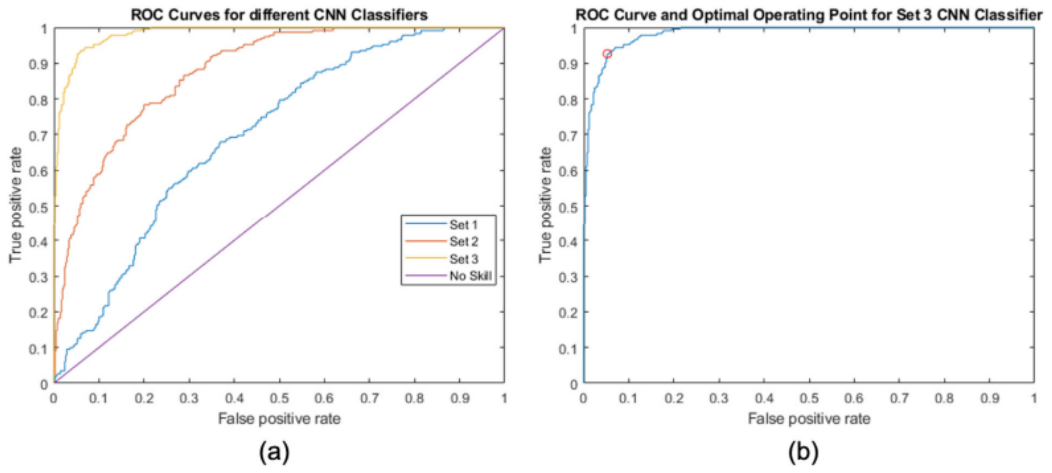


Figure 3.9: (a) Four ROC curves are drawn to compare the classifiers trained by three different sets and the no skill classifier. (b) ROC curve for the CNN classifier trained from set 3 with the red circle shows the optimal classification threshold.

CNN classifier is shown in Figure 3.9. One quantitative metric to reflect the performance of the classifier is to look at the area under the ROC curve (AUC), which provides an aggregate measure of performance across all possible classification thresholds. One interpretation of AUC is the probability that the model ranks a random positive example higher than a random negative example. A perfect classifier would have an AUC of 1.0 as it can perfectly predict all positive and negative cases correctly. On the other hand, a no-skill classifier would have an AUC of 0.5 as it randomly predicts between positive and negative classes. With these numbers as a reference, we can evaluate how well each classifier does depend on its proximity to a perfect or a no-skill classifier. As shown in Figure 3.9, the CNN trained from set 1, set 2, and set 3 has an AUC of 0.6998, 0.8774, and 0.9845 respectively.

Finally, we will discuss the applicability limits of the proposed CNN-based defect detection approach. The CNN classifier does not take into consideration of the physical properties of the DMLS process. The CNN classifier only sees the numerical values associated with the processed images and classifies them based on the gradient and intensity differences. Therefore, the pre-processing of the data into a standardized and suitable format is crucial to the success of the algorithm. For example, if we applied a different transformation for the temperature to pixel value or used a different material, such as Inconel 718, the proposed CNN will have to undergo minor adjustment and retraining to achieve similar performance. Overall, the proposed ML defect detection framework would still be applicable to material or thermal feature transformation changes.

3.3 Data Reduction

As mentioned in Section 1, one of the current challenges in scaling up DMLS manufacturing is the gigantic dataset size produced during the monitoring process. In the following section, we will connect our proposed workflow with data reduction implications and demonstrate several methods to reduce data.

3.3.1 Data Reduction in DMLS Process

The first strategy is thermal feature reduction, which refers to the transformation of the raw LWIR sensor data to the thermal feature images. The LWIR sensor first picks up light intensity signals from the DMLS build process and stores them in a time-series format. In a typical experiment example, there are seven different build parts on the build plate and each is built up to layer 240. The raw LWIR sensor data for the build plate for one layer is around 1.485 Gb which translates to around 350 Gb of data in total. Thus, for large and lengthy production processes, direct storage of the raw data is not feasible. To deal with this problem, thermal feature algorithms can be applied to avoid storing unnecessary information such as the light intensity of a location that is not currently along the laser path. For example, the maximum temperature thermal feature extracts the highest light intensity for each pixel in each layer. In the same experiment above, we can reduce 7429 frames of raw data images to a single T_{max} image for each layer, which results in around a 99.94% reduction from raw data.

Even though a large portion of the data reduction is handled by the thermal feature reduction, the CNN detection reduction is crucial when further scaling up the DMLS manufacturing process.

The main goal of the CNN detection reduction is to determine which thermal feature images to transmit for storage and which to discard. The CNN detection reduction can be separated into two parts: the CNN classifier and the transmission strategy. Adjusting the CNN classifier will affect the classification decisions made by the CNN. This is generally done by setting a classification threshold and varying this threshold to find the optimal operating point based on the importance of false positive and false negative errors. The trade-off for adjusting the CNN classifier is the performance of the classifier and the amount of data reduced. The optimal operating point based on performance is unlikely the optimal point for reducing the most data. On the other hand, adjusting the transmission strategy will affect what to do with the already classified results. The main trade-off of adjusting the transmission strategy is the amount of data reduced and the effectiveness of the data transmitted. In certain scenarios, while a large portion of data can be discarded and thus saving storage space, some key information may be lost.

3.3.2 CNN Classifier

One objective of our CNN model is to prevent unnecessary data from being transmitted and stored and this can be controlled by adjusting the CNN classifier threshold, which will vary the classifications of each image. The specific amount of data reduced is based on the transmission strategy which considers what percentage of defective and non-defective images will be transmitted to storage. Different transmission strategies will be discussed in detail in the following section. To demonstrate the effect of varying the classification threshold, we assume a naive transmission strategy of transmitting all defective images while discarding all non-defective images in this section. While other approaches may vary the amount of defective and non-defective images kept, the general trend displayed by the model performance and data reduction trade-off is the same.

Our CNN classifies the sensor images into two classes: defective (positive cases) and non-defective (negative cases). However, simply classifying the image based on the higher confidence scores of the two classes may not result in the best industrial application. For example, if an image is given a score of 55% defective and 45% non-defective, classifying this image as defective may not be the best in practice as we want to be more certain when we are categorizing an image as defective. Therefore, an absolute classification threshold should be determined based on the importance of false positive and false negative classifications. In the DMLS process, a false negative report can range from a missed hot spot on the part leading to a bad surface finish or a recoater jam that can damage the whole AM machine. A false positive report is generally considered as a false alarm and may stop the AM machine build prematurely until a process engineer is dispatched. By setting a higher classification threshold for detecting defective cases, there will be a decrease in false positive reports as it would be less inclined to classify some uncertain errors as defective. Similarly, setting a lower classification threshold will increase the amount of false positive reports while decreasing negative reports. Figure 3.10 shows the general trend of varying different thresholds and its effect on model performance and data reduction.

In order to determine the optimal operating point of the CNN classifier, we refer back to the previously drawn ROC curves in Figure 3.9. A guideline with slope S, calculated by Equation 3.8:

$$S = \frac{Cost(P|N) - Cost(N|N)}{Cost(N|P) - Cost(P|P)} * \frac{N}{P} \tag{3.8}$$

where $Cost(x|y)$ is the cost of misclassifying a y class as an x class where x and y can be any combination of P and N, N represents a negative class, and P represents a positive class. In our work, we assumed equal value of the positive and negative classes. After determining the slope of the line, we can find the optimal operating point by moving the guideline from the top left corner of the ROC plot in the direction of the bottom right corner, until it intersects the ROC curve. The optimal operating point is indicated by the red circle in Figure 3.9. To avoid confusion, only the classifier trained from set 3 is drawn as it had the highest accuracy and AUC.

3.3.3 Transmission Strategy

In the previous section, we have varied the classification threshold of the CNN classifier to examine the trade-off between model performance and data reduction. Another key aspect of

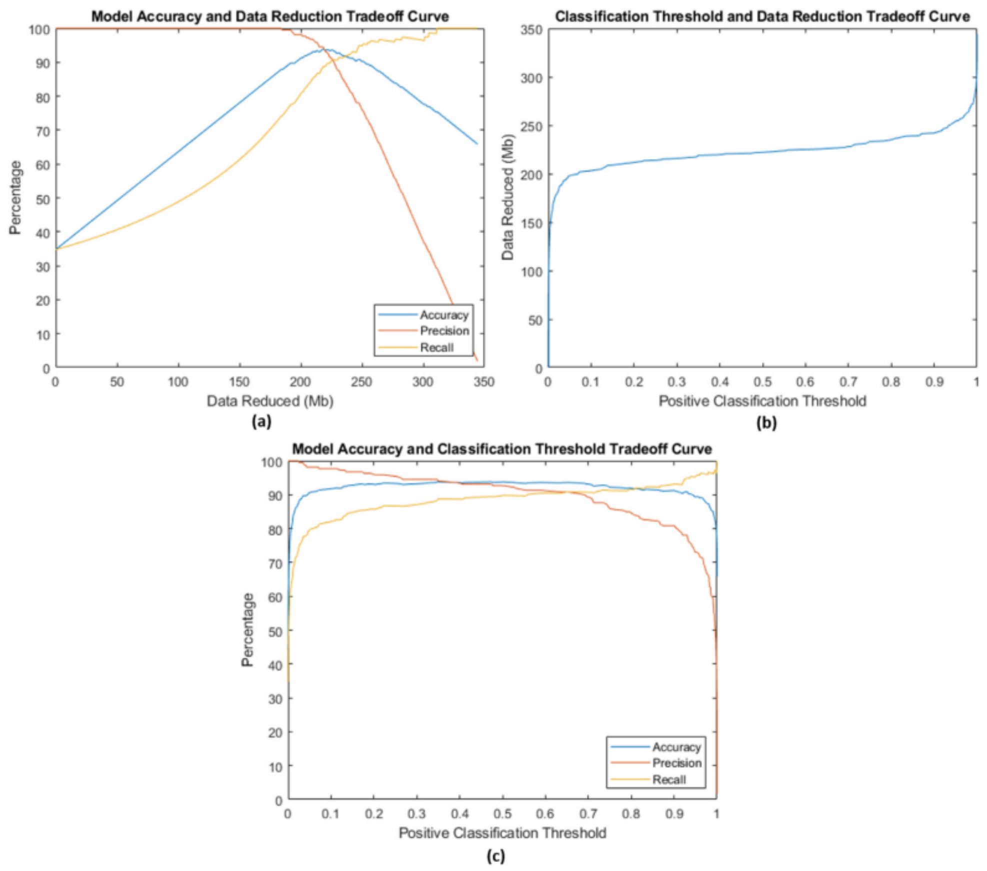


Figure 3.10: Different trade-offs between: (a) performance and data reduced, (b) data reduced and classification threshold, and (c) performance and classification threshold.

the data reduction trade-off is the transmission strategy, which considers which images should be transmitted for further analysis and training purposes. Within the total pool of sensor images, not all images are helpful for error analysis or CNN improvement. Therefore, we want to transmit this type of image at a reduced rate compared to images containing rare defects. Another goal of transmitting sensor images is to allow the further tuning of the CNN classifier and to improve its accuracy during manufacturing cycles. Later in this section, we will compare the differences between using a complete selection strategy versus a random selection versus a score-based image transmission strategy.

To describe the trade-offs between data reduction and data effectiveness, we use the retrained CNN accuracy as the metric. The higher accuracy of the retrained CNN corresponds to a higher quality of data being transmitted. Regardless of the selected transmission strategy, we take a portion, 60% in our case, of the original CNN training set and combine it with the newly transmitted images as our new training set. We still use a portion of the original CNN training dataset because we want to use it as an anchor dataset in case of dramatic disturbance shift or error. However, not the entire original training dataset is used because we want to allow the CNN to improve upon each iteration by emphasizing the importance of the incoming images, which generally corresponds to more relevant and up-to-date data. Therefore, more weight should be given to the dataset when conducting retraining.

Table 3.2 provides a summary of the different transmission strategies and their respective performance and data reduction level. With the complete selection strategy, we transmit all of the classified images based on the current cycle. This method guarantees none of the crucial information being lost but is deficient from a data reduction perspective. With the random selection strategy, we transmit a portion of classified images selected at random. Since defective images are generally more valued than non-defective images, we select a higher number of defective images than non-defective images. In our example, we selected 80% of the total defective images and only 20% of the non-defective images. This selection strategy achieves a reasonable amount of data reduction around 57%, but cannot achieve the same level of accuracy as the other

Table 3.2: Performance summary of the different transmission strategies both on a regular CNN and CNN with FEM augmentation.

	CNN	CNN with FEM Augmentation
Original (No retraining)	Accuracy: 45.3% Data reduction: n/a	Accuracy: 50%, Data reduction: n/a
Complete Transmission	Accuracy: 50.3%, Data reduction: 0%	Accuracy: 95.1%, Data reduction: 0%
Random Sampling Transmission	Accuracy: 51.4%, Data reduction: 58.3%	Accuracy: 83.1%, Data reduction: 57.4%
Score-based Transmission	Accuracy: 60.3%, Data reduction: 36.6%	Accuracy: 95.2%, Data reduction: 75.0%

strategies. The final proposed strategy is the score-based selection strategy. In our CNN predicted result, a significant portion of the classified pictures scores above the maximum score threshold, which indicates that the previous CNN iteration can already detect these types of errors. Thus, transmitting all of the images may result in inefficient use of the data storage, we can eliminate the uncertain classifications through the implementation of a minimum score threshold and also achieve data reduction through the implementation of a maximum score threshold. With the score-based transmission strategy, we are able to reduce the incoming data by 75%. As shown in Table 3.2, the proposed CNN with FEM augmentation paired with score-based transmission strategy achieves the highest accuracy as well as the most data reduction. Performance can be further improved if more detailed tuning is conducted concerning specific processes. Example tuning parameters include the previously mentioned classification threshold and the maximum and minimum score-based thresholds.

3.4 Conclusion

Several levels of FEM modeling and simulation were investigated in this work. A mesoscale model was first constructed which demonstrates that the heat transfer and melt pool behavior is successfully reconstructed with respect to experimental works and analytical relationships. Microscale tests were then investigated. In particular, the laser source simplification and heat conductivity rectification are proven to achieve significant computational demand reduction while preserving model fidelity. Based on these modeling works, a part-scale model was developed to reproduce experimentally relevant thermal feature analysis. This FEM model simulated an experimental semi-arch part geometry. In addition, the model will be useful in understanding experimental defects, and also in the CNN training data augmentation. Transfer learning was performed on a modified ResNet CNN backbone to reduce computational load. Three different CNN classifiers trained from different datasets are compared to each other in performance through examining their respective confusion matrices and ROC curves. It can be concluded that the CNN classifier augmented with FEM thermal images has shown much improved performance than its non-augmented counterpart by increasing detection accuracy from 45.2% to 92.3%. Finally, data reduction implications were discussed in relation to the CNN classifier to overcome the problem of data transmission and storage.

Chapter 4

In-Situ Thermographic Inspection for Local Powder Layer Thickness Estimation in Laser Powder Bed Fusion

The DMLS process, which is a type of laser powder bed fusion (LPBF) process, is strongly influenced by the characteristics of the powder layer, including its thickness and thermal transport properties. In this chapter, in-situ characterization of the powder layer using thermographic inspection is investigated. A thermal camera monitors the temperature history of the powder surface immediately after a layer of new powder is deposited by the recoating system. During this process, thermal energy diffuses from the underlying solid part, eventually raising the temperature of the above powder layer. Guided by 1D modeling of this heat-up process, experiments show how the parameterized thermal history can be correlated with powder layer thickness and its thermal conductivity. A neural network, based on the parameterized thermal history, further improves the correlation after training. It is used to predict the part distortion for an unsupported structure. This method detects serious part distortion several layers before the part breaks through the powder layer and interacts with the recoater. This approach can be automated to prevent catastrophic recoater crashes or abrasion of soft wipers and has the potential to monitor local properties of the powder layer in-situ.

4.1 Experimental Setup

Figure 4.1 illustrates the experimental setup with a long-wavelength infrared (LWIR) camera (FLIR A655sc) integrated with a commercial LPBF machine (Renishaw AM250). A ZnSe window on the chamber allows the LWIR camera to observe the build plate from 15° off-normal. The LWIR camera has 640 × 480 pixels which corresponds to a spatial resolution of 325 μm in this configuration. The spectral range of this LWIR camera is 7.5–14.0 μm. The noise-equivalent temperature difference (NETD) is 30 mK and frame rate is 200 Hz. The camera reports the temperature of the scene assuming an emissivity of 0.95 (gray body approximation) and is not corrected for transmission through the window. Because the emissivity of the powder and the printed solid material as well as the wiper differ, the measured temperatures are lower than the true temperatures. However, the relevant information for the procedure presented in this chapter is in the time domain. This allows the measured temperatures to be normalized for calculations and removes requirements for emissivity and nonuniformity calibrations, significantly simplifying the instrumentation. Figure 4.2 shows the temperature history of a single pixel for a 67.6 s duration layer during a typical build using 304 L stainless steel powder in a Renishaw AM250. The powder used in this chapter had diameters ranging between 15 and 40 μm with a mean diameter 25 μm. At the start of the process, powder is dispensed from the hopper at the back of the chamber. The wiper spreads the powder over the build area, traveling from the back of the chamber to the front of the chamber. The time index is shifted so that the wiper passes over the pixel in Figure 4.2 at $t = 0$. As the wiper completes its travel, it pushes any excess powder into a catchment at the front of the chamber. The laser is then scanned over the pattern for the layer to fuse the powder to the part. The camera captures this interaction but is saturated when the pixel is heated directly by the laser. At the conclusion of the laser patterning, the build plate is lowered by the layer height, δ, and the wiper returns to the back of the chamber, passing back over the pixel at $t = 63.5s$. The emissivity of the pixel changes depending on the states of the material and presence of the wiper. These changes are indicated by the color of the trace.

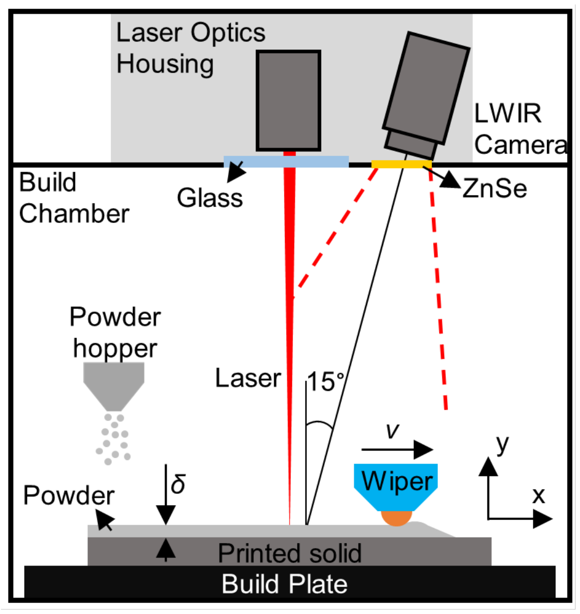


Figure 4.1: Schematic of LPBF experimental setup with LWIR camera.

Figure 4.2(b) shows a close-up of the interaction with the wiper during the powder-spreading step. Initially the pixel corresponds to the printed part which has a lower emissivity. This has cooled from the interaction with the laser, although there still retains some residual heat as well as any heating from the build plate (maintained at $80°C$ for the builds). The wiper pushes powder in front of it. Some of this powder enters the pixel before the wiper. Although the fresh powder is cooler, the increased emissivity leads to a slight increase in the measured temperature to T_{min} at $T = 60.75°C$. When the wiper blocks the camera's view of the pixel there is a decrease in the measured temperature (indicated in grey), followed by a sharp rise in the measured temperature ($t = 0$). The surface temperature of the powder increases as heat diffuses from the underlying part. This asymptotically approaches a steady state value of T_{max}, which is the temperature of the build plate i.e. $T_{max} = T_s = 80°C$. It should be noted that both T_{min} and T_{max} have the same emissivity because they both correspond to the powder state. The dynamic response of the surface temperature is a function of the powder layer thickness as well as the thermal properties of the powder. The remainder of the chapter focuses on this portion of the temperaturetime history. It is convenient to normalize the local surface temperature by

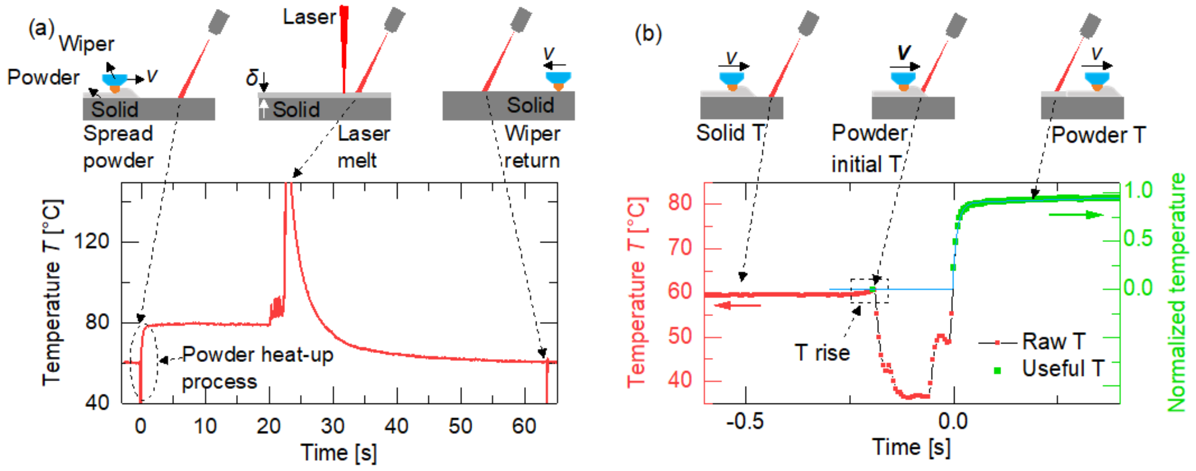


Figure 4.2: Schematic of powder spreading process in temperature history of (a) one complete layer (b) powder heat-up portion.

$$\frac{T - T_{min}}{T_{max} - T_{min}} \tag{4.1}$$

which removes the effects of the temperature of the underlying part as well as the dependence on the camera's calibration.

4.2 1D heat transfer model and finite differential elements solution

4.2.1 1D Heat Transfer Model

Insight into the temperature history can be achieved from modeling the thermal diffusion through the powder from the underlying part. While the powder layer is made of discrete particles, evaluating it as continuous effective medium significantly simplifies the analysis. Also, the thickness of the powder is much less than the lateral extent of most parts, so it is reasonable to model the problem as 1D problem in the z-direction. Lastly, the temperature range for the system during the recoating step does not vary significantly enough to consider thermally dependent properties. Figure 4.3(a) illustrates this model. A powder layer of thickness, δ, is in contact with the printed solid part. Radiation exchange at the free surface of the powder is neglected, but a convection boundary condition is applied. The printed solid part is modeled as semi-infinite. The temperature of the semi-infinite end side is constant temperature at $T_s = 80°C$. This problem is governed by the heat transfer equation,

$$\frac{\partial T}{\partial t} = \frac{k_i}{\rho_i c_i} \frac{\partial^2 T}{\partial z^2} \tag{4.2}$$

with boundary conditions,

$$k_p \frac{\partial T}{\partial z} = h(T(z=0,t) - T_\infty) \tag{4.3}$$

$$T(z \to \infty, t) = T_s \tag{4.4}$$

and initial condition,

$$T(z, t=0) = \begin{cases} T_p & 0 < z \leq \delta \\ T_s & \delta < z \end{cases} \tag{4.5}$$

where T is the temperature and is a function of time, t, and distance from the surface of the powder, z. The thermal properties in Equation 4.2 depend on the material and can be either powder or printed solid (denoted by the subscript, i, which is p for powder and s for printed solid). k, ρ and c are the effective thermal conductivity, density and specific heat of the materials. h is the convection heat transfer coefficient. T_∞ is the ambient temperature while T_s and T_p are the initial temperatures of the printed solid and powder, respectively.

The powder's effective thermal conductivity is fit from experiments as described in Section 4.2. The calculated temperature distribution is shown as a function of depth z in Figure 3(b) for $T_s = 80°C, T_p = 60°C, T_\infty = 60°C$, $h = 15W/m^2K$. The thermal camera only has access to the surface temperature of the powder. The model predicts that the powder surface temperature approaches the steady state within almost 1s.

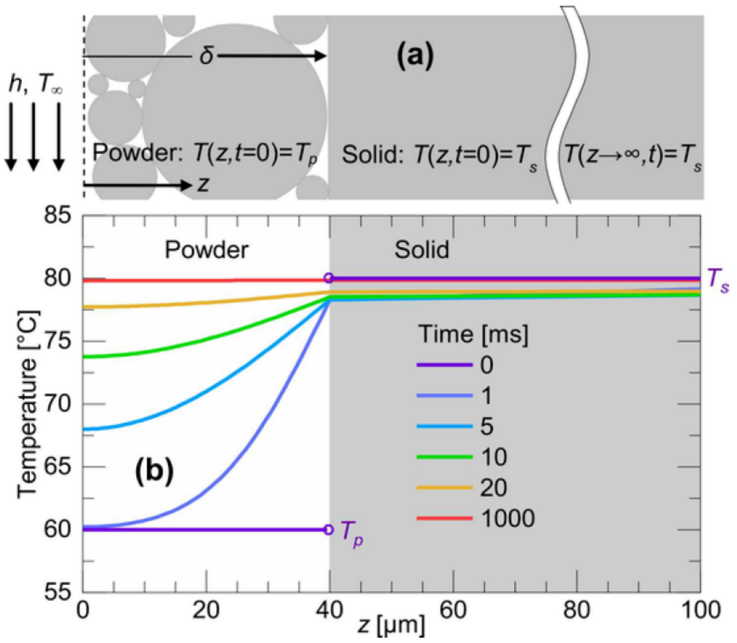


Figure 4.3: 1D heat diffusion model (a) model definition and (b) simulated temperature distribution at various times.

4.2.2 Thermal Feature: Rise Time

It is helpful to have a parameterized expression to fit to the surface temperature history. The following equation captures the surface temperature rise predicted by the finite difference solution to Equations 4.2–4.5.

$$T(z=0,t) = A_1 - A_2 e^{\beta_1 (t-t_i)} - A_3 e^{\beta_2 (t-t_i)} \tag{4.6}$$

where A_1 is the powder free surface's steady-state temperature and $A_1 - A_2 - A_3$ is the powder free surface's initial temperature, T_p. Replacing t with $t - t_i$ allows the time history to be shifted so the initial condition occurs at t_i (when the wiper introduces fresh powder to the printed solid surface). The coefficients β_1 and β_2 depend on the powder layer thickness and thermal diffusivity of the materials. For a sufficiently thick powder layer on a high thermal diffusivity material $(\alpha = k/\rho c)$ material. While this serves as a starting point, it breaks down for thinner powder layers and this chapter fits the coefficients numerically for both simulation and experiments. After normalization using Equation 4.1, the initial temperature of powder free surface $T\prime(z=0, t=t_i)$ is zero, so that

$A_1 - A_2 - A_3 = 0$ and Equation 4.6 can be simplified as

$$T'(z=0,t) = A_1 - A_2 e^{\beta_1 (t-t_i)} - (A_1 - A_2) e^{\beta_2 (t-t_i)} \tag{4.7}$$

The coefficients $A_1, A_2, \beta_1, \beta_2$ and t_i can be fit to the free surface temperature. Figure 4.4(a) shows the ability of Equation 4.7 to capture the free surface temperature predicted by the finite difference model for different powder layer thicknesses. The mean squared error for the powder thicknesses in Figure 4.4(a) is less than 5.3×10^{-6}.

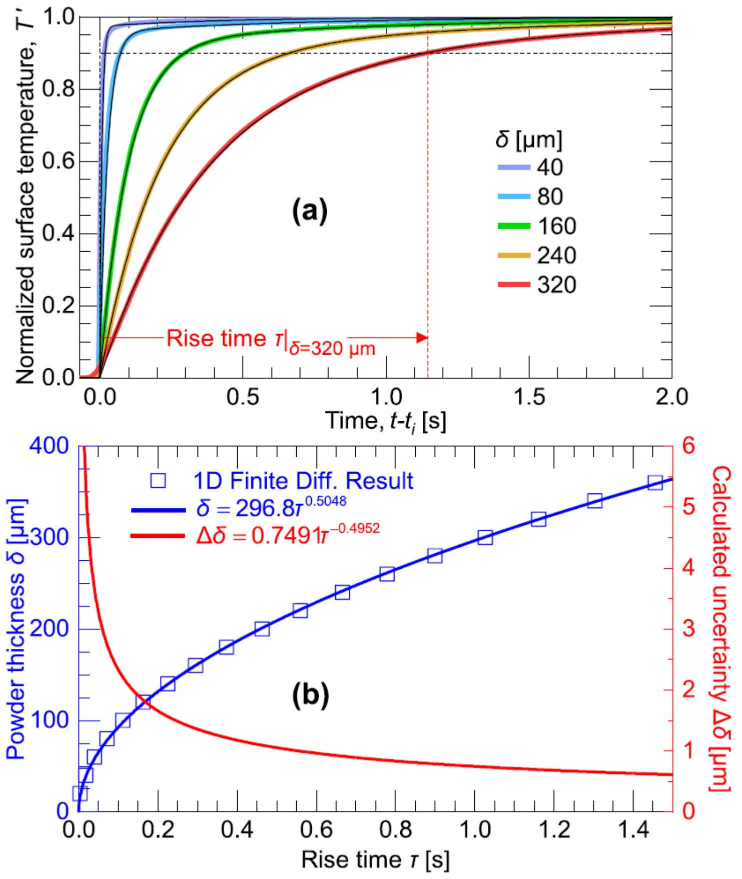


Figure 4.4: Fitted 1D thermal diffusion model results (a) parameterized fitted results overlayed on results from finite difference solution with rise time for the δ = 320 μm layer highlighted (b) powder thickness as a function of rise time with power law fit and calculated powder layer thickness uncertainty.

A characteristic rise time, τ, defined as the time from $t - t_i$ for the surface temperature to reach 90% of steady state, can be obtained from Equation 4.7.

$$A_2 e^{\alpha_1 \tau} + (A_1 - A_2) e^{\alpha_2 \tau} = 0.1 A_1 \tag{4.8}$$

The rise time serves as a thermal feature which can be related to the powder layer thickness or the thermal properties. Figure 4(b) shows the dependence of the rise time on the powder layer thickness from finite difference model calculated results. This relationship is captured by a power law dependence

$$\delta = 269.8 \tau^{0.5048} \tag{4.9}$$

Assuming the constant properties in Table 1, this function can be used to predict the powder thickness δ from rise-time τ, which can be fit to simulated or experimental data. The uncertainty in the predicted powder layer thickness depends on the first derivative of Equation 4.9. An estimate for the uncertainty in the powder layer thickness, $\Delta\delta$ is given by

$$\Delta\delta = \frac{\partial \delta}{\partial \tau} \Delta\tau \tag{4.10}$$

Table 4.1: Material properties of 304 L solid and powder.

Properties	304 L Solid	304 L Powder
Density $[kg/m^3]$	8030	4818
Specific heat capacity $[J/kgK]$	490	490
Thermal conductivity $[W/mK]$	16.2	0.269

where $\Delta\tau$ is the uncertainty in the rise time. For example, if we assume that the uncertainty in the time constant scales with the reciprocal of the frame rate of the camera (200 Hz), the uncertainty for a 40 μm powder layer would be 5.21 μm while the uncertainty for a 100 μm decreases to 2.17 μm. Coincidentally, this agrees with the breakdown in the treatment of the powder layer as an effective medium as the layer thickness approaches the powder particle diameter (40 μm for the powder in this chapter).

4.3 Experimental Validation

The parametric expressions in Equations 4.7 and 4.8 can also be applied to experimentally gathered data, and the experimentally fit coefficients can then be used to predict the powder thickness. This approach requires collection of temperature histories for various known powder thicknesses. Figure 4.5(a) and b shows a simple gage specimen printed with the Renishaw AM250. It consists of 5 *mm* wide strips, 35 *mm* long, at different elevations h from a printed datum plane. After the specimen was printed, the unfused powder is removed from above the gage specimen. The specimen is not removed from the build platform. After the powder is removed the chamber is resealed again, the argon atmosphere restored, and the build plate heated to $80°C$, before commanding the system to add a new layer of powder. The wiper then spreads powder over the build plate including the specimen. This gives a powder layer thickness that varies spatially with the height of the individual strips of the gage specimen. The powder layer thickness is $\delta = h_0 - h$, where h_0 is the height above the datum of the build plane. In this experiment h_0 = 360 μm so that the highest specimen has δ = 40 μm of powder above it. The heights of the gage specimen and corresponding powder thicknesses are listed in Table 2. The 8 different heights in the table are repeated three times along the width of the specimen. The thermal histories for each pixel above the specimen are recorded during the recoating process. After this, the build plate is removed from chamber and cleaned. The top surface of the specimen is then laser scanned (LMI Gocater 2320) to measure precisely and map the height of each strip above the datum plane. Figure 5(b) shows the measured height of each specimen and Figure 5(c) shows a spatial map of the corresponding powder thickness registered to the camera pixels. This shows minimal local variances from the nominal powder thickness.

Figure 6(a) shows examples of the temperature histories for pixels with different nominal powder thicknesses. Equation 4.7 is fit to the temperature history pixel-by-pixel. The five fitted

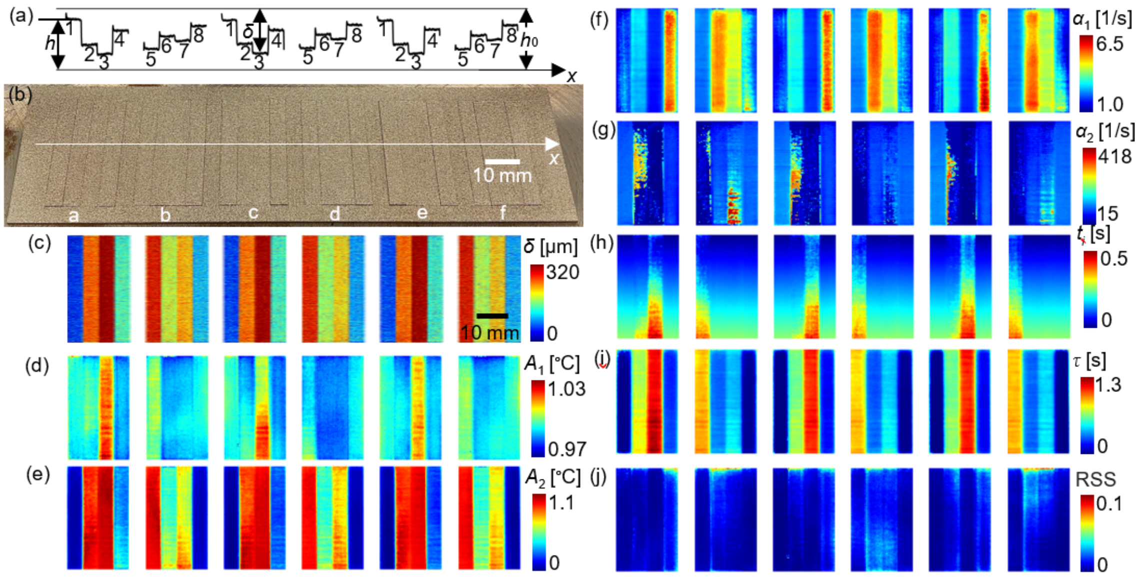


Figure 4.5: (a) Photograph of height artifact and (b) surface profile along cross section A-A. 2D maps of the (c) measured powder layer thickness, (d) rise-time τ, (e) A_1, (f) A_2, (g) β_1, (h) β_2, (i) t_i, (j) coefficient of determination 2.

Table 4.2: The height of each of the eight different strips and corresponding powder thickness δ.

n	h [μm]	δ [μm]
1	320	40
2	120	240
3	40	320
4	240	120
5	80	280
6	200	160
7	160	200
8	280	80

coefficients are plotted in Figure 4.5(e)-(i). As mentioned in Section 3.2, A_1 corresponds to the normalized steady state temperature and should have a nominal value of unity. Figure 4.5(e) shows that the fitted values of A_1 range between 0.97 and 1.03. The time shift t_i is related to when the wiper passes over the pixel. The coefficient β_1 is inversely proportionate to the thickness of the powder layer as should be expected. Figure 4.5(f) shows that t_i scales linearly with y for thinner powder layers. This agrees with the constant velocity ($v = 140mm/s$) of the wiper in the

y-direction. For thicker powder layers, there is a delay between when the powder is deposited and when a temperature rise can be observed on the free surface because of the time it takes for thermal energy to diffuse through the powder layer.

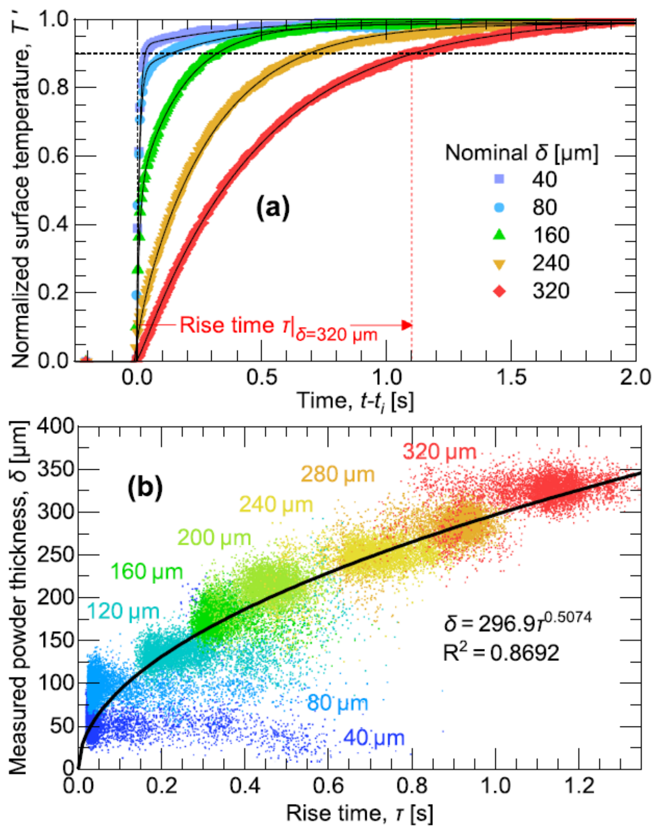


Figure 4.6: (a) Measured temperature histories for different powder thicknesses and fitted parameters (b) powder thickness δ with respect to rise-time τ.

The coefficient of determination, R^2, for these fits is shown in Figure 4.5(j). The figure shows that Equation 4.7 captures the local temperature histories well and the minimum R^2 over the entire gage specimen is 0.98 with an average for all the data of $R^2 = 0.997$. The five fitted coefficients can be combined using Equation 4.8 to calculate the rise time for each pixel. This is plotted in Figure 4.5(d). The correspondence between powder layer thickness and rise time is clearly visible in Figure 4.5(c) and (d). Figure 4.6(b) also shows the powder layer thickness as a function of rise time. A power law fit to this data gives the relationship

$$\delta = 269.9\tau^{0.5074} \tag{4.11}$$

which has $R^2 = 0.869$.

While the parameterization from the 1D model captures the response well for the interior of the strips as illustrated in Figure 4.6(a). Figure 4.6(b) shows significant scatter in the rise time and the measured powder thickness. Much of this occurs at the boundary between strips. In these regions, the assumptions of the 1D model are not satisfied and the thermal disturbance spreads laterally (cylindrically away from the edge defined by the adjacent strips). Experimentally, this lateral spread is on the order of two pixels (0.65 *mm* for this setup) and is convoluted with resolution of the camera.

4.4 Machine Learning with FNN Regression Learner for Powder Thickness Prediction

The previous section showed that working with the rise time leads to straightforward correlations to the powder thickness or thermal properties. However, the relatively low R^2 in Figure 4.6(b) indicates a large variance of predictive performance in certain regions. Specifically, for powder thicknesses $\delta < 180\mu m$, the power law model shows a large variance as certain points are over-estimated by as much as over 200%. In application, this large error will cause the prediction algorithm to miss defects at low powder thickness levels since the algorithm tends to over-estimate powder thickness levels.

The consolidation of the five thermal history fitted parameters (A_1,A_2,β_1,β_2, and t_i) fitted thermal history to the rise time can lead to some loss of information. Generating physics based correlations with all five parameters is challenging. Machine learning techniques are well suited to this type of problem, provided sufficient data is available. A feedforward neural network (FNN) is applied to the data in Figure 4.5 using the Regression Learner toolbox in MATLAB. The five parameters A_1,A_2,β_1,β_2, and t_i in Figure 4.5(e)-(i) are directly used to train the FNN to predict the powder layer thickness.

The general structure of an FNN model can be mathematically represented by the following equations

$$a_k^{[1]} = \sigma^{[l]}(\sum_{i=1}^{n^{[1]}} x_i w_{ij}^{[1]} + b^{[1]}) \tag{4.12a}$$

$$a_k^{[l-1]} = \sigma^{[l-1]}(\sum_{i=1}^{n^{[l-1]}} a_i^{[l-2]} w_{ij}^{[l-1]} + b^{[l-1]}) \tag{4.12b}$$

$$y_j = \sigma^{[l]}(\sum_{i=1}^{n^{[l]}} a_i^{[l-1]} w_{ij}^{[l]} + b^{[l]}) \tag{4.12c}$$

where x_i is input features and y_j is the desired output target. is the weight from neuron i of the $k-1$ layer to neuron j of the k layer. l and $n^{[k]}$ denote the number of layers and number of neurons for the k layer, respectively. $b^{[k]}$ and $\sigma^{[k]}$ are the bias and activation function for the k layer.

In this study, there are 42510 total data points that are split based on a ratio of 70–15–15 into the training, validation, and testing dataset for the FNN, respectively. The training dataset is used to adjust the weights and biases of the FNN, the validation dataset is used to adjust the hyperparameters (number neurons) of the FNN, and the testing dataset is used to evaluate the performance of the FNN to an unseen dataset. It is important to note that data splitting is conducted before any further data processing steps such as normalization and scaling to prevent data leakage. Using the regression learner MATLAB toolbox, several different FNN architectures are tested against each other with their own specifications. A model with a single hidden layer with 100 hidden neurons and ReLu as the activation function displayed the best performance with an R-squared value of 0.962.

Figure 4.7 compares the prediction results from the rise time only correlation and the five parameters based FNN. The FNN preforms significantly better for thinner powder layers. The overall coefficient of determination R^2 improves to 0.962 for the FNN as opposed to 0.859 using the rise time correlation. Figure 4.7(c) shows histograms of the true powder thicknesses when the predicted powder layer thickness is $\delta_p = 200 \pm 5\mu m$. Fitting a normal distribution to the results from both models gives a 95% confidence interval estimate of the true powder layer thickness of $\delta = 194.2 \pm 60.6\mu m$ for the rise time correlation which is improved to $\delta = 201.5 \pm 32.2\mu m$ for the FNN model.

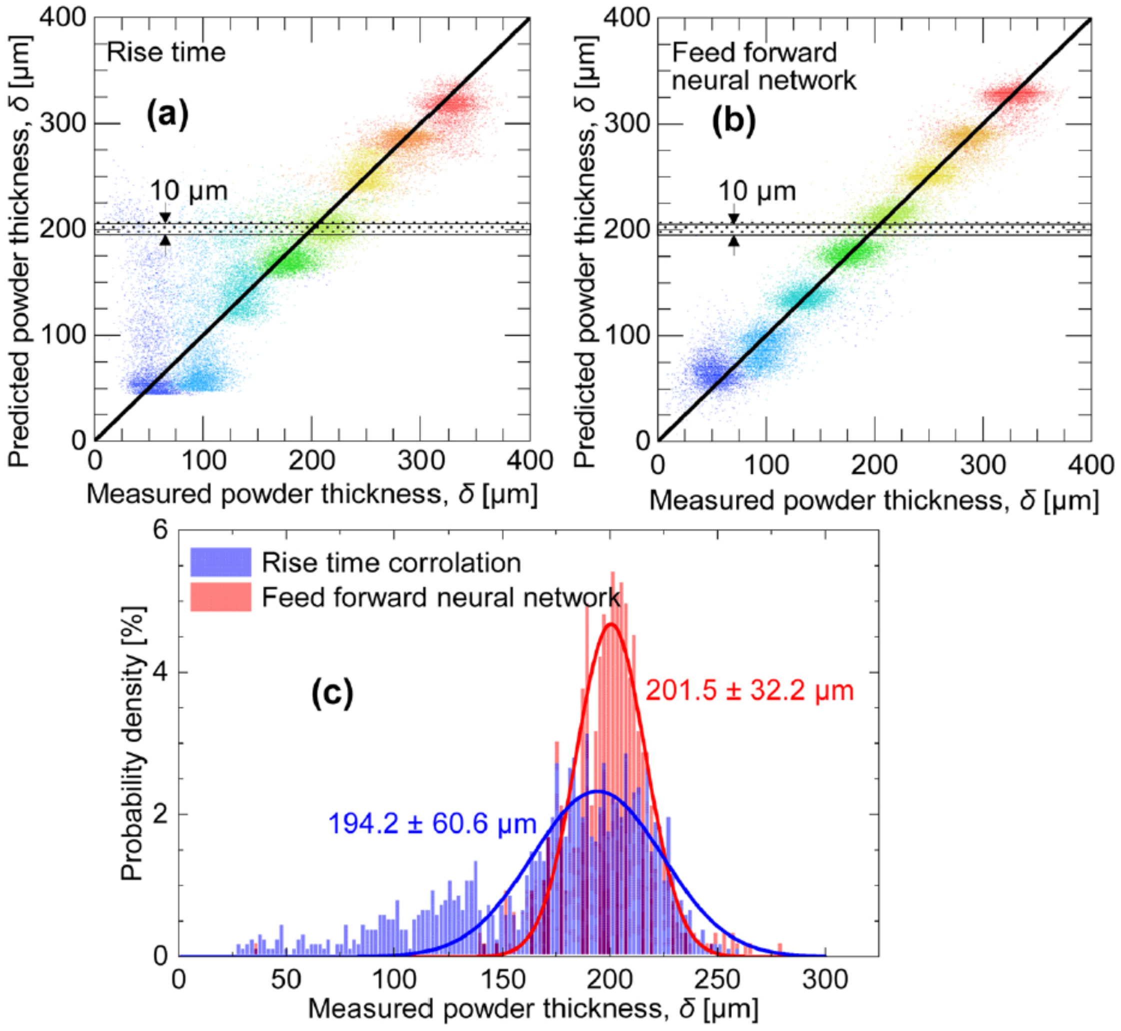


Figure 4.7: Predicted powder thickness δ by using (a) rise time method, (b) neural network regression method and (c) powder thickness with normal distribution when the predicted powder layer thickness is $p = 200 \pm 5\ \mu m$.

4.5 In-situ Thermographic Monitoring for Recoater Interaction

Thermal distortion occurs commonly in LPBF. This can occur for unsupported parts when there is inadequate conduction to the build plate leading to excessive heating. The distortion lowers the powder thickness on next layer which exasperates the overheating as less material is melted. When the distortion of the part causes it to break through the powder layer, it can interact with the recoater. The interaction of the recoater with the part can abrade a soft wiper while a rigid wiper can crash into the part and damage the machine, part, or both. In the case of a flexible wiper, the abraded area allows an increase in the powder layer thickness for every part in-line with the abraded area. The contact with the wiper is potentially catastrophic for both types of wipers. Detecting thermal distortion prior to a part breaking through the powder layer allows the local process parameters to be adjusted in order to limit distortion or to have the part suppressed to save the build. The distorted.part leads to a change in the local powder layer thickness which can be detected using the thermographic powder layer monitoring technique.

Artifacts with an overhang are printed to demonstrate a change in powder layer thickness caused by thermal stress as reported in our previous work [134]. The dimensions of the overhang artifact are shown in Figure 4.8(a). Figure 4.8(b) shows a photograph of the specimen and a block printed with the Renishaw AM250 system using a flexible (silicone) wiper. The block is added behind the specimen to capture any subsequent changes in the powder bed thickness due to the abraded wiper. These artifacts are printed using nominal process parameters developed for stainless steel 304 L. Specifically, a laser power of 200 W, beam size of 75 μm, scan speed of 800 mm/s and setup nominal powder thickness $\delta_N = 50\mu m$. For simplicity, the laser was only rastered in the xdirection. The parts in Figure 4.8(b) were stopped after 219 layers and show significant distortion in the unsupported overhanging region. The height of the powder layer is monitored using the thermal camera, the temperature history for each point fitted using the 5-parameters in Figure 5, and the FNN used to predict the powder layer thickness. Figure 4.9(a) shows the predicted powder thickness averaged along the leading edge of the overhang artifact (red line). For reference, the predicted powder thickness averaged over a line at the center of each layer (blue line) is also plotted. The constant raster direction leads to a slightly higher part (lower thickness) at the edge almost immediately. This is stable by layer 40. The powder layer thickness on the unsupported edge starts decreasing more and more at layer 180 due to thermal distortion. Over the entire print, the reference thickness in the center of the part is stable.

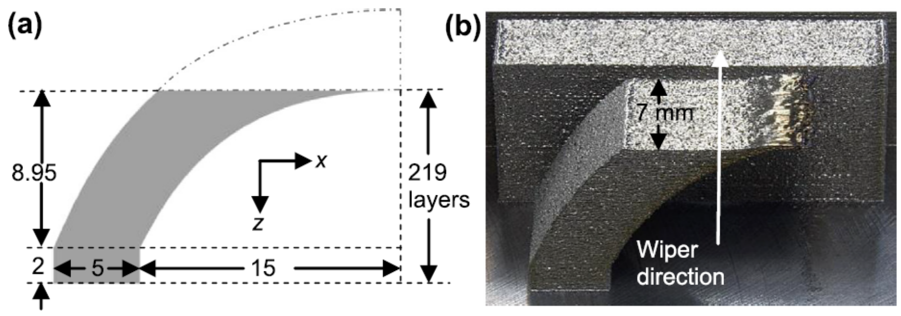


Figure 4.8: (a) Schematic image of overhang with dimensions in mm (b) photograph of distortion grows on the overhang top surface with a block behind it.

To verify these predictions, specimens were stopped at intermediate heights during the build. Photographs of the specimens stopped at layers 50, 140, and 215 are shown in Figure 4.9(a). At the conclusion of the build, these parts were laser scanned (after removing powder but without removing the parts from the build plate). The nominal height is calculated by multiplying the number of layers with the nominal powder layer thickness. The actual powder layer thickness is this nominal height minus the laser scanner measured height. This thickness is plotted for both the unsupported edge and center of the part in Figure 4.9(a) as points, by averaging the data measured over each line (red dash line or blue dash line) for the various layers. This validation was repeated for three different parts (stopped at the same layer) and the error bars show the standard deviations. The figure shows that the predicted layer height from the thermal inspection method agrees well with the measured height. In particular, the thermal distortion predicted that the part broke through the powder layer at layer 212 while the measurements show a broke-through between layers 200 and 205.

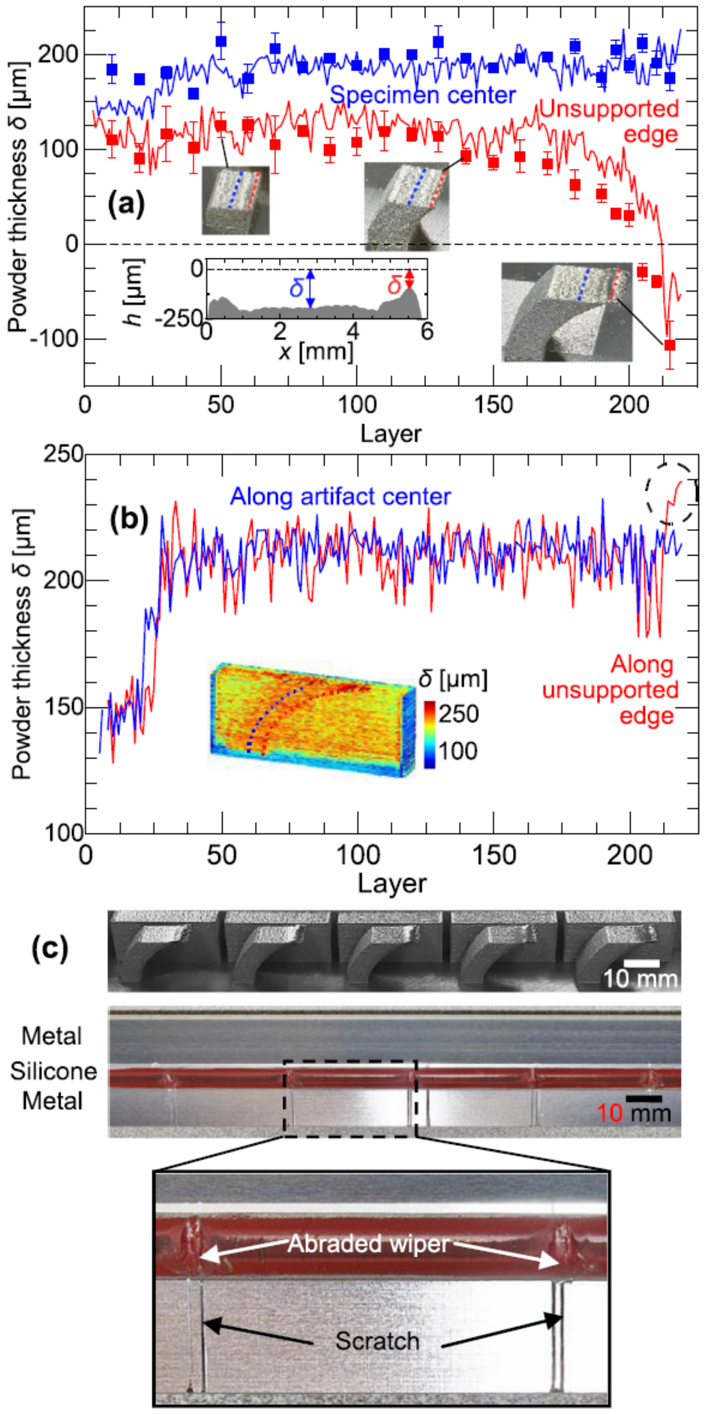


Figure 4.9: (a) Comparison of powder thickness δ between thermal measure and height measure. The blue line indicates the inner center powder thickness averaged over the blue dots line for every layer. The red line denotes the unsupported edge powder thickness averaged over the red dots line for every layer. (b) Powder thickness δ at different block areas. Blue and red lines represent the powder thickness of block along blue and red dots with respect to layers. The insert images are the 3D map of block powder thickness δ obtained from the thermal measure. (c) Final overhang samples and the abraded wiper.

Figure 4.9(b) shows the thermal inspection predicted height for the block behind the overhang artifact. The predicted powder layer height is plotted for the same *x*-position as the traces in Figure 4.9(a) (immediately behind the leading edge of the artifact and at its center). The powder layer stabilizes by layer 40. At layer 215, the local powder layer thickness behind the unsupported edge starts to increase dramatically. This is due to the abrasion of the wiper and leads to the local deposition of more powder. While not visible in the photograph of the part in Figure 4.8(b), if this continued, it could negatively affect the local part properties.

Figure 4.9(c) shows photographs of five identical parts, stopped at layer 219, and the wiper. The damage to the wiper is visible (scratched metal and abraded silicone). The decrease in the powder layer thickness indicating an imminent crash could be detected by layer 190 from the thermal inspection estimate. This is at least 10 layers before the collision and would allow the part to be suppressed to prevent damage to the wiper as well as any other parts in the build.

It is interesting to observe that the measured as well as predicted powder layer thicknesses are so much larger than the nominal powder layer thickness, about 200 and 50 μm, respectively. This is consistent with [105] who showed a nominal powder thickness of 20 μm resulted in a measured powder layer thickness of more than 100 μm as well as [166] who showed nominal powder layer thicknesses of 30 and 50 μm corresponded to actual thicknesses of 165 and 225 μm. This phenomenon is the result of shrinkage of powder consolidation over multiple layers [105] in addition to significant losses of the powder from spatter and denudation [166].

In addition to suppressing a bad part before it breaks through, detecting thermal distortion early could be used to adjust the process parameters to match the reduced powder layer thickness. This includes lowering the laser power or increasing the scan speed. There is also the possibility to adjust the scan path to minimize overheating. Because the distortion is detected early, there is time to react by making more gradual changes. Other more drastic approaches are possible, including laser ablation of the distorted region or physically deforming it down.

4.6 Conclusion

This chapter demonstrates the potential of using in-situ thermographic inspection to evaluate the powder layer thickness and estimate local powder layer thermal properties. An experimental study, conducted using realistic process parameters in a commercial LPBF machine, agrees well with a 1D thermal diffusion model for the powder layer. This uses a stationary LWIR thermal camera in a staring configuration. A parametric fit of the temperature history is used to generate a correlation to the powder layer thickness. A simplified rise time model provides insight and was used to estimate thermal conductivity, but it was outperformed by a FNN machine learning method for data obtained from a calibration specimen. This was tested for a simple artifact with an overhanging unsupported edge. Without underlying supported structure, the ability of the overhang to dissipate heat is significantly reduced, causing heat to build up at the edge. Overheating causes thermal distortion, which reduces the thickness of the next powder layer. The overheating is exasperated as there is less powder for the laser to melt and the distortion rapidly increases to the point where it breaks through the powder layer and produces the abrasion of the silicone wiper. The FNN model was applied to monitor thermal distortion and could ultimately predict a wiper collision at least 10 layers before it occurred.

The thermal inspection method appears to have several advantages relative to existing methods for monitoring the powder layer in-situ. In particular, because the inspection occurs during the recoating step, it does not add additional processing time for the inspection. Comparing with optical coherence tomography and fringe projection methods, this method's additional equipment costs are minimal. The thermographic inspection technique was able to resolve the powder layer height to $\pm$ 32.2 μm (within the mean particle diameter).

Chapter 5

A three-level hierachical framework for additive manufacturing

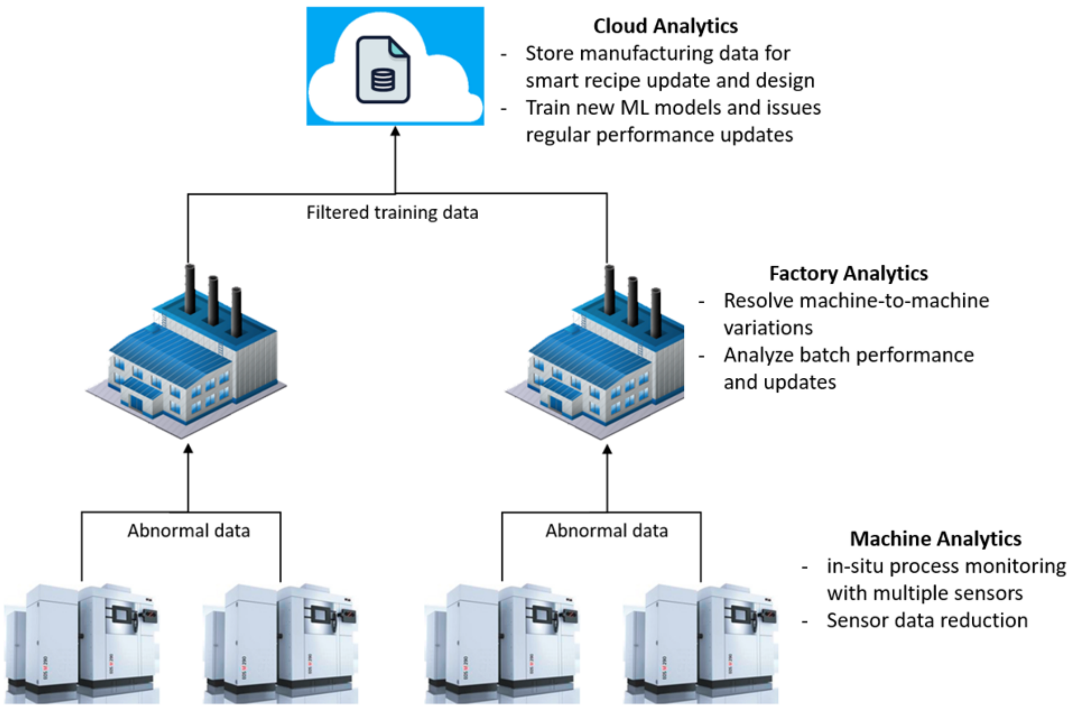
Metal alloy additive manufacturing (AM) has gained wide industrial interest in the past decade due to its capability to efficiently deliver complicated mechanical parts with high quality. However, due to a lack of understanding of the fundamental correlation between the operating conditions and build quality, the exploration of the optimal operating policy of the AM process is costly and difficult. In this chapter, a data-driven process optimization framework has been proposed for the additive manufacturing process, integrating machine learning, finite-element method (FEM) modeling, and cloud-edge data storage/transfer optimization. A three-level hierarchy of local machines, factory clouds, and a research center is introduced with each level responsible for its dedicated tasks. In addition, to ensure the efficiency of data transfer and storage, an edge-cloud data transfer scheme is constructed, which serves as a guideline for the data flow in the AM framework. Moreover, an overview of the connections between the proposed framework and the Industry 4.0 framework is presented.

5.1 AM Framework Hierarchy

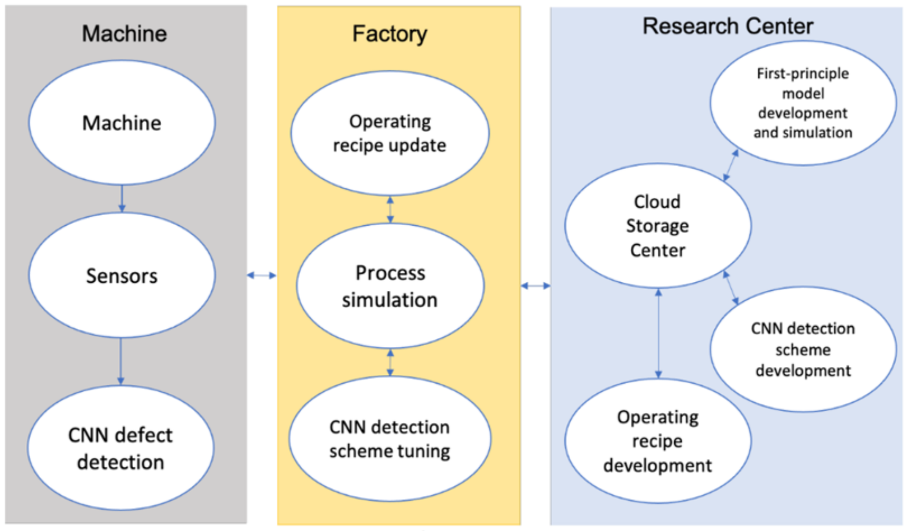
The following sections discuss the proposed additive manufacturing framework, which is constituted of three levels: the machine level, the factory level and the research center level. An overview of the roles and responsibilities of each level is demonstrated in Figure 5.1.

5.1.1 Machine

The lowest level of the additive manufacturing framework hierarchy is the machine level, where build parts are produced and manufacturing data are constantly generated and collected. An AM machine, e.g. EOS M290, receives the build recipe from the factory level and makes the build with the received recipe. A typical additive manufacturing recipe includes, but does not limit to, laser scanning path, hatch spacing, laser power, and powder thickness. Depending on the build geometry and material, the building parameters can vary dramatically, causing the building process to range from a couple of hours to a day. Due to a lack of an efficient real-time model that predicts the build part details online, it is crucial to monitor the AM process with the appropriate sensors to prevent and understand undesirable defects. The adoption of multiple sensors to provide better manufacturing results is prevalent in the additive manufacturing process. For example, [177] developed a multi-sensor framework for the wire arc additive manufacturing, where individual sensors contribute differently to the final decision making. In addition, [44] also designed a multi-sensor in-situ monitoring system for the AM process. For the LPBF process, two of the most commonly used sensors to monitor the in-situ heat transfer aspect of the process, based on their functionalities, are the optical tomography (OT) sensor and melt pool (MP) sensor [54]. The EOS M290 machine uses the EOS Suite, a process monitoring software, to perform the online monitoring using the information from these sensors. Both sensors can provide a unique perspective on the AM process. Specifically, the OT sensor outputs a whole image of the powder bed that summarizes the maximal temperature that occurred on the platform throughout the build process. The OT sensor is an off-axis sensor, which does not move along the laser path and


Cloud Analytics
- Store manufacturing data for smart recipe update and design
- Train new ML models and issues regular performance updates
Filtered training data
Factory Analytics
- Resolve machine-to-machine variations
- Analyze batch performance and updates
Abnormal data
Abnormal data
Machine Analytics
- in-situ process monitoring with multiple sensors
- Sensor data reduction

(a)


Machine
Machine
Sensors
CNN defect detection
Factory
Operating recipe update
Process simulation
CNN detection scheme tuning
Research Center
First-principle model development and simulation
Cloud Storage Center
CNN detection scheme development
Operating recipe development

(b)

Figure 5.1: An overview of the components of the additive manufacturing framework and their respective responsibilities. (a) A hierarchical depiction of the AM framework. (b) The machine level is shown in the gray block, the factory level is shown in the yellow block and the research center level is shown in the blue block.

in-sensor post-processing is necessary to adjust for the optical distortion due to different view angles. Additionally, since the OT sensor captures the entire build plate, the overall resolution will be lower. On the other hand, the MP sensor captures the local image of the melt pool. Since the MP sensor is an on-axis camera and aims to observe the transient behavior of the melt pool, it generates images at a much higher frequency than the OT sensor. Since the MP sensor focuses on a much smaller area, it produces a higher resolution over the local melt pool region. In other types of machines, e.g. Renishaw AM250, the EOS monitor suite is unavailable, but analogous cameras can be installed to achieve a similar effect. Specifically, [100] have installed a short-wave infrared (SWIR) camera on a Renishaw M250 machine to monitor the in-situ melt pool behavior of the process, similarly to the MP sensor. Experiments are also be done with thermal feature processing with the long-wave infrared (LWIR) camera to achieve similar results to the OT sensor.

5.1.1.1 Machine Sensor Monitoring

Even though sensor results are interpretable to process engineers, to further automate the defect identification process, it is desirable to adopt techniques that make the sensor results machine-readable. Moreover, the raw data size from sensors can be extremely large, reaching up to 30 GB per build. Therefore, it is also important to perform efficient data reduction on the machine level to ensure the efficiency of data transfer speed and data storage size. Since AM sensor results are more easily interpreted as images, the raw time-series sensor data is first processed and converted into a layer-wise image of the build plate. For example, the thermal temperature data from an LWIR camera is transformed into a thermal mapping of the build plate through a thermal feature extraction algorithm. To classify these processed sensor images, CNNs have been widely adopted as an image classification technique and their accuracy has been acclaimed by a variety of industries [89]. CNN can extract and recognize important features within an image in a highly efficient manner. Therefore, we propose to utilize CNN to process and interpret the AM sensor results. In this framework, each sensor is proposed to pair with a dedicated and specifically-trained CNN to perform the defect identification task on the local machine. The extracted features allow automated error detection and recipe update, which also allows redundant information to be discarded and keeps only essential information. It is also noteworthy that, a deployed CNN is relatively fast at execution and does not require a large amount of computation power to run. Thus, even if the machine-level computers may not have the strongest hardware, these computers are adequately equipped to perform the image classification task. [134] has proposed a CNN defect detection workflow and tested using experimental data. The proposed CNN workflow can classify images at high accuracy and timely manner.

5.1.1.2 Sensor Cross Validation and Machine Transmission Details

Different sensors, such as the OT, MP, or powder-bed sensors, provide different perspectives on the manufacturing process. Therefore, it is important to weigh the sensor results based on the intrinsic strengths of each sensor. [135] has shown that the OT sensor performs better at identifying the over-melting problem, while the MP sensor performs better at identifying the under-melting problem. Similarly, the powder-bed sensor can also be cross-validated against the OT or MP sensors to identify problems such as recoater jams. Therefore, to fully take advantage of the strengths of the sensors, a cross-validation scheme is proposed to select the most reasonable sensor results using statistical analysis.

A crucial aspect of the proposed AM framework at the machine level is the information that is transmitted from machine to factory. CNNs and other process monitoring workflows can be implemented to reduce the amount of data transmitted. This filtered process information should be stored in an efficient and ready for use format before being transmitted to the next hierarchy of the framework, the factory level. In the following section, a cross-validation scheme between different sensors and the data transfer format is proposed for the example of two types of errors, e_1 and e_2, monitored by two different sensors, S_1 and S_2.

First, the problematic areas within each build are identified and their locations and error types are collected using the dedicated sensor CNNs,

$$[(e_1,l)_j,(e_2,l)_j] = CNN_i(I_i), i \in \{S_1,S_2\} \tag{5.1}$$

where CNN$_i$ is the trained network for a specific sensor i, I_i is the sensor raw data, i is the type of sensor image, e_j is the likelihood of a potential error label, l_j is the error location, $j \in [0,N_e]$ is the error index, and N_e is the total number of errors. Additionally, the output dimension is $N_e \times 2$ due to the two types of sensors involved in this case.

For one particular location l_j, it will have two types of error confidence levels reported by two types of sensors,

$$\begin{aligned}(e_1,l_j)_{final} &= \lambda_{S_1,1}(e_1,l_j)_{S_1} + \lambda_{S_2,1}(e_1,l_j)_{S_2}\\(e_2,l_j)_{final} &= \lambda_{S_1,2}(e_2,l_j)_{S_1} + \lambda_{S_2,2}(e_2,l_j)_{S_2}\end{aligned} \tag{5.2}$$

where $(e_i,l_j)_{final}$ is the overall likelihood of an error type i at location l_j and $\lambda_{k,i}$ is the weight determined by cross-validation for sensor k.

Then a filter is applied to every location to determine the most likely error. Confidence levels below a certain threshold will be regarded as error-free.

$$\begin{aligned}&E = \Big[filter\Big((e_1,l_j)_{final},(e_2,l_j)_{final}\Big)\Big]\ j \in [0,N_e']\\&filter: \{\text{Pre-filter } e \text{ with a threshold; Then apply } \max(\cdot,\cdot)\}\end{aligned} \tag{5.3}$$

where E is the cross-validated error list.

Although the example provided here uses two sensors and detects two error types, the proposed cross validation scheme can also be potentially generalized to more sensors and more types of errors. In addition, human knowledge can also be used for the determination of errors and these aspects can be explored in future works which focus more on the detailed implementation of the proposed framework.

5.1.2 Factory

As mentioned in the previous section, the factory level is a local computation cluster with moderate computing power and its physical location would ideally be in the AM factory to ensure efficient communication with the AM machines. The role of the factory level is to serve as a checkpoint and coordinator of the AM machines. When the pre-trained CNNs on the AM machine detect some potential defects, the error information, as well as the corresponding operating policies, will be sent to the factory for quality assurance and recipe correction if possible. Process engineers will be stationed at the factory level to perform quality control on incoming data. In the event of an error, process engineers first can utilize the local resources at the factory level including the local recipe book and simulation center in an attempt to find solutions to the error. Process engineers can transmit the error and its corresponding build information to the research center if further investigation is needed. In the case of misclassification of incoming data by the CNN algorithm, process engineers can selectively transmit the necessary data to the research center to improve and update the CNN algorithm.

5.1.2.1 CNN Deployment Monitoring

One of the most important tasks at the factory level is the quality assurance of the manufacturing process by skilled process engineers. At the machine level, it is not realistic to have process engineers monitor the various sensor results of each individual machine, hence the implementation of CNN algorithm to pre-filter out results. However, at the factory level, it is important for process engineers to regularly perform quality control on the machine products. While the CNN algorithm on each machine should already be performing at an acceptable high level, it still is susceptible to unexpected events and manufacturing data drift. For example, during the life cycle of an AM machine, the sensor calibration may shift and result in a systematic error in data distribution. If the new distribution is beyond the implemented range of the trained CNN network, then the CNN classification results will not be accurate and potentially impact the product quality. Examples of other unexpected events include sensor failures and faulty machines, which are also beyond the knowledge of the original CNN algorithm. Process engineers will monitor subsets of the incoming data from the machine level and respond to any potential events following the appropriate protocol. In addition, another benefit in having human experts monitor the results of the CNN data is the opportunity to improve CNN performance. If an incoming dataset contains incorrect classifications, the process engineers can selectively transmit a subset of the incoming images with the corrected labels to the research center for updates to the CNN algorithm. Detailed transmission schemes and a case study of this proposed workflow will be explained further in Section 5.2.

5.1.2.2 RNN Workflow

On the factory level, a hybrid recipe update scheme is proposed which aims to provide smart recipe correction through the combined effort of both a high-level recurrent neural network (RNN) and human experts. A recurrent network is often used to predict outcomes based on historical data [110]. Since the machine-level manufacturing data are essentially time sequences of thermal and structural information, they can be processed by an RNN using a high-level analysis. Therefore, at the factory level, a pre-trained high-level RNN can be used as an operating recipe encyclopedia. The operating encyclopedia contains a large manufacturing dataset that can be used to derive the correlation between the operating conditions and potential defects. Therefore, the RNN can provide suggestions in the adjustment of AM operating policies when defects are detected. However, in the early phase of the manufacturing of a new build geometry, the manufacturing data are usually insufficient for the training of a meaningful RNN. Therefore, human knowledge can be extremely helpful in assisting the adjustment of AM recipes. In addition to that, human knowledge can also be applied to efficiently select the dataset to train the RNN with higher accuracy. However, with the collection of more manufacturing data, the feedback process can further be automated, thereby, reducing human intervention.

As mentioned in Section 5.1.1.2, the proposed recipe-update scheme is implemented using the transmitted error list, E, from individual local machines. Due to the standardized format of the transmitted data, the RNN-based workflow can be directly implemented without too much pre-processing. The formulation of the RNN network is shown as follows,

$$u' = RNN(E, b, u)$$

$$RNN: f(h(t), m(t), u(t), [e, l], b) = h(t+L), m(t+L), u'(t+L)$$

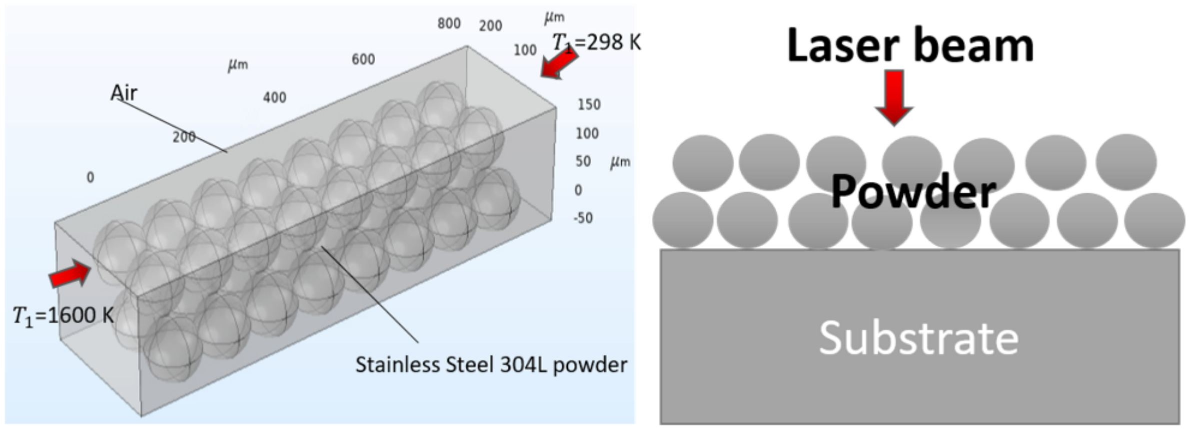
where h is a hidden state and m is a memory state. Using the error list and the additional build history around errors b, from an individual machine, the RNN can predict an updated recipe u' that can potentially fix the defects during the production. The same RNN network can be directly applied for multiple local machines and resolve the errors of the same type. It is noteworthy that, in the beginning phase of the manufacturing, it is difficult to derive a good RNN network due to the limited amount of manufacturing data. Therefore, human expert knowledge can also be adopted to facilitate the recipe enhancement.

5.1.2.3 Simulation Center

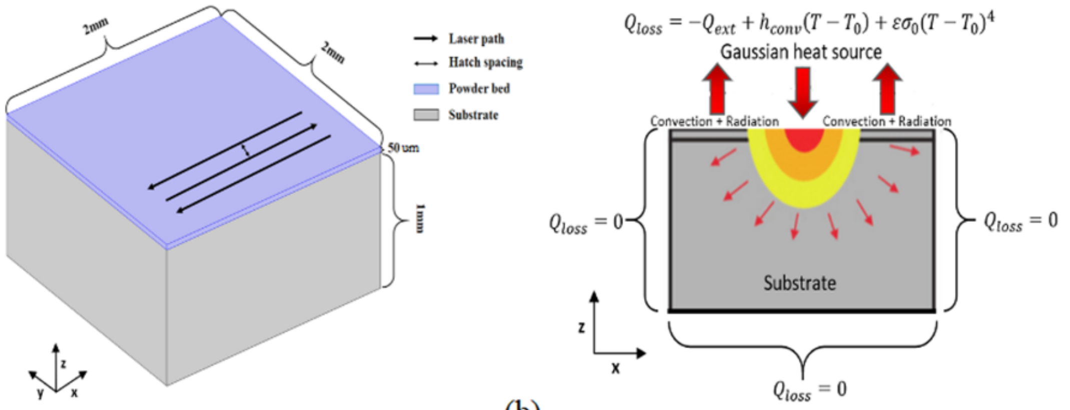
The other role of the factory is to serve as a local AM process simulation center. The computation power of the factory cluster can be used to perform the solution of a pre-built FEM model simulation that does not require too much computational resource. These simulations, such as elementary structural and thermal analysis, can provide additional information regarding printing processes and avoid the usage of ex-situ analysis, which is often destructive to the printing product. Even with the limited computation power on the factory cloud, many types of simulations can be performed. [135] has explored the possible simulations that can be performed on the factory level. According to the domain and scale that simulation models focus on, three types of simulations can be conducted on the factory level: micro-scale, meso-scale, and part-scale. An example of a micro-scale simulation is shown in Figure 5.2(a). The shown 1-D microscopic transport model can be used to explore the thermodynamic and transport properties of the materials that are manufactured. In this particular case, the built 1-D model is used to compute the effective thermal conductivity of the powder material through numerical experiments. These types of simulations can be used as a useful supplement to the parameter determination studies, such as described in [51]. For meso-scale simulation, an example is shown in Figure 5.2(b). The meso-scale simulation focuses on a portion of the built part, which can be a feature of interest or a sub-part. By constructing an FEM model, the local thermal history details of the part can be extracted for further analysis. The meso-scale simulation is particularly useful to obtain insights into the defects or distortions that occur on a specific geometric feature or location. Finally, part-scale simulation captures the dynamics of the entire build part or even a build plate where multiple parts are being built. An exemplary part-scale simulation geometry is shown in Figure 5.2(c). The part-scale simulation can provide a holistic perspective of the building process, and it can be used to validate the building recipe and prevent part defects through simulation prediction. All three types of simulations are suitable for the computation resources available on a factory cluster. Additionally, due to the resemblance of part-scale simulation results with the sensor data, part-scale simulations can also be used as a data-augmentation method to further enhance the robustness of recipe design, which will be covered in more detail in the next section.

5.1.3 Research Center

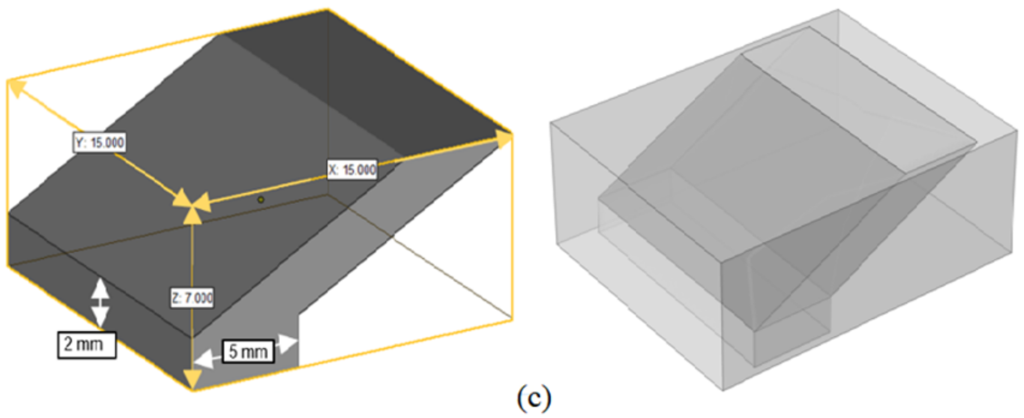
The highest level in the hierarchy of the framework is the research center, which can be assumed to be a supercomputer cluster equipped with strong computation power. Computation resources are typically limited on local venues due to the lack of sufficient hardware infrastructure and high cost. Therefore, cloud computing has been proposed as an alternative to providing strong computational power to terminal users in a shared environment. The cloud computing market has witnessed significant growth in the past decade, and the technologies involved with cloud computing have matured to be able to meet the computation demand by the global research and commercial activities nowadays [133]. Many cloud computing resources are available commercially, of which the most representative one is the AWS cloud service provided by Amazon [13]. In this framework, such a cloud computing hub is used as a dedicated research center for the AM process, which may serve and oversee hundreds of factories. With the strong computing power of the research



(a)



(b)



(c)

Figure 5.2: Three examples of suitable simulations that can be run at the factory level. (a) Micro-scale simulation on the powder properties. (b) Meso-scale simulation on a portion of the build part. (c) Part-scale simulation of the entire build part.

center, tasks that are not feasible on the local machine and factory cloud can be performed here. It is noteworthy that the research center does not directly take part in the manufacturing process, and does not make recommendations for the recipe online. Instead, the research center serves as a knowledge hub that constantly takes in selected manufacturing data and performs machine learning studies, such as the training of a high-level recurrent neural network model, to provide the next-generation recipe for process optimization.

The research center can perform more computationally demanding simulations. For example, on the factory level, only one FEM model can be executed to validate a certain range of operating conditions due to the limitation of computational capacity. However, at the research center, a parametric sweep can be performed to inspect a large range of operating conditions, thus providing much more insights into the process. The type of simulations that can be performed at the research center also has a larger variety than at the factory cloud level. For instance, detailed simulation of the microscopic properties through the first-principles-based method can be performed, which is too computationally expensive on the factory cloud or local machine. For example, density functional theory calculations can be performed to obtain the necessary thermodynamic properties of the build material through the efficient approximate solution of the Schrödinger's equation [68]. These first-principles-based simulation results, along with the large amount of manufacturing data provided by different machines on different build parts can provide sufficient quantity and generality of the additive manufacturing process. With these data, a robust model can be trained to provide insights on recipe designs for new build parts. Last but not least, another role of the research center is data storage. The stable infrastructure of supercomputer clusters makes the research center the ideal place for the storage and easy access of manufacturing data, which is typically on the scale of petabytes or even exabytes. However, as mentioned before, the research center is expected to receive data from hundreds and thousands of AM machines from the local factories. Therefore, the large amount of data influx and data transfer may be overwhelming even for a supercomputer like the research center. As a result, Section 5.2 proposes an edge-cloud data transfer scheme that can potentially optimize the data transfer and storage.

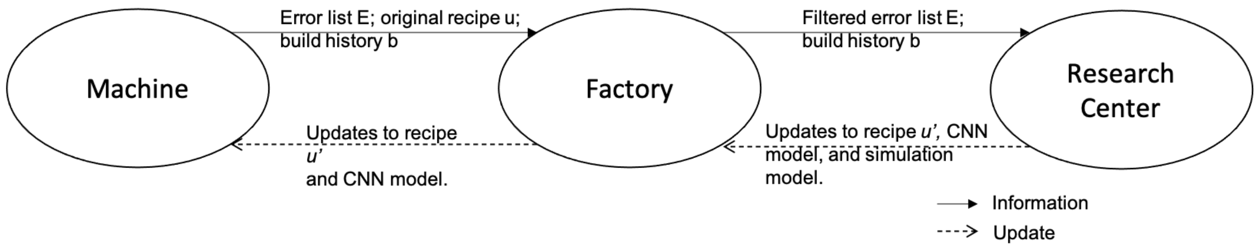

Machine
Factory
Research Center
Error list E; original recipe u; build history b
Filtered error list E; build history b
Updates to recipe u' and CNN model.
Updates to recipe u', CNN model, and simulation model.
Information
Update

Figure 5.3: An overview of the edge-cloud data transfer scheme of the additive manufacturing framework. The arrow orientations represent the directions of the data flow.

With the enormous amount of data gathered from different factories, the research center can also serve as a decision-making hub for the industry as a whole. [26] argued that enterprise-wide profitability is reliant on a number of integrated factors such as reliability, safety, flexibility, and environmental concerns. As discussed in Section 5.1.1.2, in-situ process monitoring of AM machines is an example of reliability and safety factors. The integration of ML techniques will allow more flexibility and turn-over rate on recipe design experiments through techniques such as AI-surrogate models and digital twins [21]. Environmental concerns may include the design of the input powder material and geometric shapes. It is important to allow consistent information flow between different factors as it contributes to aligning the enterprise-wide short-term milestones with long-term milestones. Following this approach, a number of short-term and long-term goals can be established with the large and expansive range of data collected in the research center.

5.2 Edge-Cloud Data Transfer Details

As mentioned in Section 5.1.3, due to the large size of the AM manufacturing data, it is crucial to perform data reduction and maximize the data transfer efficiency. The overall data transfer flow occurring in the AM workflow is shown in Figure 5.3. In summary, the data transfer procedure is proposed to be the following: first, an initial build recipe is provided to the AM machine by the factory cloud. As the machine builds the part, the sensor data, I, are processed with CNN, and an error list E is generated. The error list is further processed through the cross-validation scheme and a cross-validated error list, E, is generated. After the data processing has been completed, the data transfer is initiated from the local machine to both the factory cloud and the research center. The data size of the cross-validated error list, E, is estimated to be about several Megabytes per transfer and the data size of the raw sensor data, I, is estimated to be about several Gigabytes per transfer. Due to the large data size of the raw sensor data, not all data should be transmitted to the research center for every build. Instead, a data transfer frequency should be determined to ensure data completeness, data transfer bandwidth occupation, and data storage efficiency. To ensure the efficiency in data storage and transfer, the following data selection and transfer paradigm is proposed based on if a part defect or recipe error is observed:

```
Machine → Factory
if error present then
    if new error then
        Factory → Research Center at $f_1$
    else
        Factory → Research Center at $f_2$% if an old error is persistent
    end
else
    Factory → Research Center at $f_3$
end
where $f_1, f_2, f_3$ are different data transfer frequencies; $f_2 > f_1 > f_3$
```

Specifically, in the proposed paradigm, the data transfers are all initiated in a linear fashion from the local machine to the factory cloud and finally to the research center to prevent data duplication. During manufacturing, if an error or defect is detected by the machine monitoring system and has not been encountered before, the machine will upload the manufacturing data to the factory cloud for error amelioration. After the data is sent to the factory cloud, according to a pre-trained recipe update scheme, an updated recipe will be deployed to the local machine, attempting to mitigate the error. Besides the feedback to the local machine, if the error has never been encountered before, i.e., a new error, the data will be also sent to the research center for bookkeeping and further study at a frequency of f_1. On the other hand, if the detected error or defect has occurred before and keeps reoccurring despite the recipe update from the factory cloud, the data will be sent to the research center from the factory cloud and the manufacturing should be halted if the severity of the defect is high. This data transfer will happen at a frequency of f_2. In the final case, suppose there is no error encountered during the manufacturing, the manufacturing data will still be transferred to the research center for routine backup. However, the frequency of routine backup, f_3, should be much slower than f_1 and f_2. The actual choice of data transfer frequencies is at plant designers' discretion, but an overall guideline for the priority of data transfer would be $f_2 > f_1 > f_3$.

5.2.1 Data Transfer Case Study

Dataset used in this study is the same dataset used in the author's previous work [134]. In summary, the dataset consists of processed LWIR camera data from different semi-arch geometries. The dataset is split into training, validation, and testing and the testing dataset is from a more recent build than the training and validation dataset. In a typical factory setting, the CNN classifier installed on each machine should already be performing at a relatively high level. The CNN used is able to achieve an accuracy of 92.3% on a fairly balanced dataset of defective and non-defective images. The testing dataset will be used to represent a new set of build information from a typical AM machine. In this case study, the CNN classifier will be evaluated before and after transmitting the testing images and updating the CNN classifier.

In this case study, we will be investigating the effectiveness of the proposed framework, specifically, the transmission between factory and cloud. As mentioned at the factory level, the incoming images will first be classified in an AI-assisted manner in which process engineers will conduct quality control on the CNN classifications. First, we manually classify the new images to recreate the job of the process engineers who will monitor the incoming images. The new set of images are grouped into two categories: images of which the CNN-determined classifications that agree with human expert knowledge, and images of which the classifications that do not agree with human knowledge. As we are dealing with classifying unstructured data, images, the human classification, or the human-level performance, can be thought of as a good ground truth classification. Ideally, we want to minimize the total data transmission size while maximizing the effectiveness of the transmitted data. Therefore, these two categories of incoming images should be transmitted at different rates. Then, we will retrain the CNN classifier with both the new images and a subset of the original dataset. This is similar to the transmission from the factory to the cloud and the cloud retraining process. Finally, we will evaluate the performance of the newly updated CNN similarly to manufacturing ML deployment monitoring. The updated CNN is retrained five times to ensure performance consistency as there are slight variations between each training since the specific training images differ from training to training due to uniform selection. In addition, different weight initialization methods and stochastic gradient descent may result in slightly different results from time to time. Therefore, we reported both the maximum and average performance at each transmission rate in our result.

Before the retraining of the CNN, the CNN can achieve an accuracy of 91.3%. This will be our baseline to which we will compare our updated CNN. As shown in Figure 5.4 (a), we have varied the fraction of correctly classified images transmitted for retraining to investigate its effect on CNN performance. It is noteworthy that since we retrained our CNN with only 60% of the original training dataset plus the new dataset, at fractions below 0.1, the updated CNN performs worse than the original. This decrease is to be expected since we did not use 40% of the original training dataset causing some defect types to be missed. However, after merely using more than 0.1 of the new dataset, the updated CNN starts to perform much better. In our case study, the performance of the CNN stabilizes after around 0.4 of the total correctly classified images are transmitted for retraining. This supports our proposed framework of only needing to transmit a portion of all incoming images for retraining as the original CNN can already classify this subset of images correctly. However, it is still necessary to transmit them for retraining to keep up with the data drift and routine backup during manufacturing. In Figure 5.4 (b), we have varied the fraction of incorrectly classified images to CNN performance. In our case study, the performance of the CNN continues to increase until around 0.85. This fraction is significantly larger than the previous

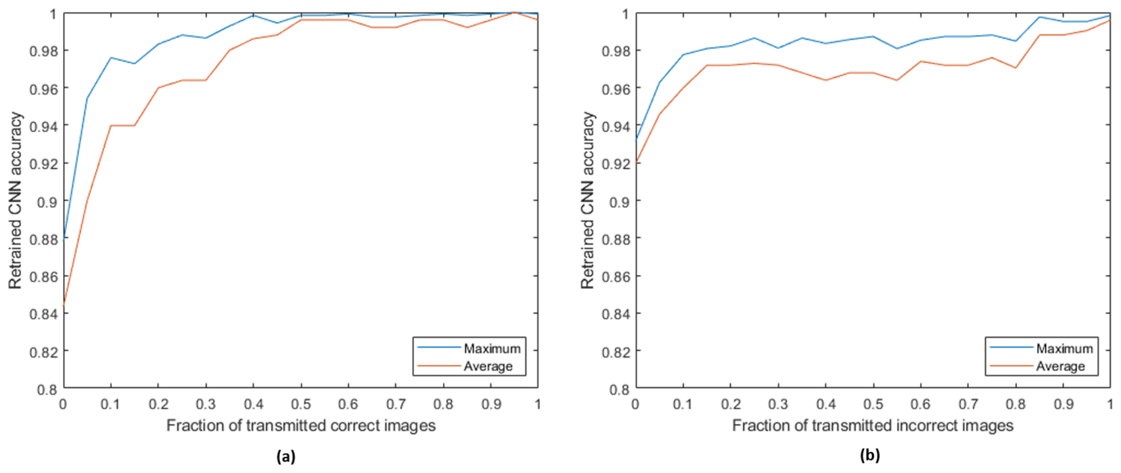


Figure 5.4: Fraction of transmitted (a) correct and (b) incorrect images and their effect on the retrained CNN accuracy.

0.4 due to this subset being the incorrectly classified images. Specifically, we varied the fraction of correctly classified images first where all of the incorrect images are sent. After, we selected the best performing fraction, 0.4, and held it constant when varying the number of incorrectly classified images. We performed the study in this order because we know that correctly classified images contribute less than incorrectly classified images in the process of retraining. We did not vary both fractions at the same time since that increases the computation cost from $O(N+M)$ to $O(N*M)$. We do not need to transmit all of the images since some of the defect types are repeated, and sending 85% of the incorrect images may be enough for the new CNN to recognize such a defect. In the case of a persistent error type throughout multiple iterations of the retraining process, this type of error should be transferred at a higher frequency or be assigned a higher weight when retraining the CNN. It is notable that the exact hyperparameters, such as transmission fractions and dataset ratios, are specific for our set of images and can vary when applied to another dataset. However, the general trends should be consistent throughout any dataset and in agreement with our proposed data transfer framework.

5.2.2 Job Scheduling

The task of data transfer in AM highly resembles the work for the implementation of the aforementioned data transfer scheme. To address this task, one can take advantage of the various network load balancing schemes and job scheduling libraries available both open-source and commercially. [122] investigated the usage of the user datagram protocol (UDP). For example, the transmission control protocol and the internet protocol (TCP/IP) or open systems interconnection (OSI) in the field of networking is a great reference for the implementation of the data transfer scheme [88, 194]. In the TCP/IP or OSI models, the data being transferred over the internet is divided into packets that are equal in size, which allows a convenient data integrity check and the usage of standardized processing protocol [59, 153]. On the other hand, the sun grid engine (SGE) is an example of the job scheduling library. A job scheduler like SGE oversees a job scheduling queue and handles the job distribution and execution based on both job priority and resource availability. When performing the job scheduling, the job scheduler optimizes and balances the job load on all available computation nodes and ensures no node gets overcrowded or empty under various circumstances [63]. For the data transfer in the AM framework, due to the large data size of the manufacturing data, customization may be necessary when using existing software or libraries.

5.3 Additive Manufacturing and Industry 4.0

Additive manufacturing is often mentioned together with the concept of Industry 4.0, which represents the fourth industrial revolution. The key idea of Industry 4.0 is the integration of smart manufacturing and information technologies into the traditional manufacturing process, which is largely driven by the desire to combine digital and physical applications, to provide efficient product customization and to promote automation in manufacturing [29, 158]. Industry 4.0 typically features the usage and development of self-aware and self-learning machines, and cloud-based manufacturing and smart manufacturing are two of the main drivers of Industry 4.0. As a result, additive manufacturing is a key component of the Industry 4.0 due to its superior prototyping capability and close relationship with other components of the Industry 4.0 [111]. [26] explained the interrelationships between additive manufacturing and cyber security, simulation, internet of things (IoT), and big data analytics.

Though still relatively not-well-developed, additive manufacturing is a broad field that has many sub-categories [167]. Despite the large variety of materials that can be manufactured by AM, the materials that are most relevant with the framework and pathway of industrial 4.0 include metallic materials, smart materials (e.g., shape memory alloys and shape memory polymers), printable hydraulics and electronics and special materials & applications (e.g., jewelry, clothing and food) [46]. Although the additive manufacturing data-driven framework proposed in this work focuses on the metal additive manufacturing and is based on the powder bed fusion (PBF) process, it can be tailored and adapted to the AM of other materials with the proper selection of sensors and potential transfer learning of the trained neural network. Moreover, the proposed data-transfer scheme is general-purpose and can be applied and customized for any additive manufacturing process that is data-intensive. In addition to the framework proposed in this work, there also exist other attempts that have been made to relate additive manufacturing with the Industry 4.0 framework, from which this work draws inspiration. For example, [161] has investigated the possibility of an IoT-enabled cloud-based additive manufacturing platform, where machine-learning techniques and hybrid human knowledge are used for the facilitation of rapid manufacturing. On the other hand, [25] has looked into the automation and the smart process-improvement of additive manufacturing through cloud computing and optimization. Moreover, recently [106] developed a big data-driven framework to assist the development of the smart AM process through big data utilization. [17] also provided a comprehensive overview of the integration of cloud computing and additive manufacturing. Therefore, it can be seen that additive manufacturing is a key element in the Industry 4.0 framework, which can potentially revolutionize the existing business models and manufacturing decisions [62, 80, 92].

Finally, an emerging field within Industry 4.0 is the integration and handling of cybersecurity concerns. When calculations, involved in the solution of process models, and/or when data analysis takes place in the factory cloud, the issue of cybersecurity should be carefully addressed as it is a problem that is occurring more and more in a variety of industries. To this end, cyber-attack detection and mitigation strategies should be implemented at the factory cloud to achieve early detection of potential cyber-attacks; a detailed treatment is outside the scope of the present work and the reader may refer to [170]'s book for an in-depth study on the modeling, detection and mitigation of cyber-attacks.

5.4 Conclusion

In this work, a data-driven process optimization framework dedicated to the additive manufacturing (AM) process is proposed. The framework is based on a three-level hierarchy: machine level, factory level, and research center level. The machine level is the lowermost level, consisting of the AM machine or the computer equipped with the AM machine. The machine level is responsible for online data collection and real-time image processing. The second level is the factory level. This level is responsible for adjusting the recipe with a predefined recipe-update policy according to the defects reported by machines. In addition, the factory level can also coordinate data transfer and perform low computationally demanding simulation tasks to provide insights on the parts that are processed at the factory level. The research center level is used as the storage hub for the manufacturing data collected from the lower levels, i.e., factory and machine levels. Using the collected manufacturing data, the research center, consisting of a supercomputer with tremendous computational resources, can perform efficient and powerful data mining to improve and develop new process recipes. Moreover, the research center can perform first-principles-based simulations that further explore the material properties to facilitate the accurate characterization of the manufacturing process. In addition to the simulation/workflow framework proposed in this work, due to the large size typically involved with the AM manufacturing data, a data-transmission scheme is also proposed to serve as a data transfer guideline for the AM process, and a case study is presented with the data scale suitable for a typical AM application. Finally, the connection between AM and the Industry 4.0 framework is elaborated to provide further motivation for the research in this field. Future work can be conducted to explore the efficient implementation of the framework proposed in this work and the integration between AM and Industry 4.0 concepts.

Chapter 6

A Tutorial Review of Neural Network Modeling Approaches for Model Predictive Control

6.1 Introduction

Model predictive control (MPC) has attracted significant research interest as it is one of the major achievements in the development of advanced multivariate process control systems due to its ability to compute the optimal control actions based on not only the instantaneous state measurements but also the anticipated process response. Specifically, MPC relies on its built-in linear/nonlinear model to capture the dynamic behavior of the process and predict the response of the process over a finite horizon window, such that it can determine the optimal control trajectory by solving a dynamic optimization problem subject to input and state constraints at every sampling time. Attributed to the achievement of developing self-tuning controllers in the 1970s, such as minimum variance (MV), generalized minimum variance (GMV) [37,38], and Pole Placement [163] controls, the methodology of MPC was proposed. Since its conceptualization, MPC has been developed and modified in various ways over the past few decades. [132] provided a tutorial review of MPC approaches for control practitioners and the variables/parameters that must be considered when designing an MPC. [108] carried out an exhaustive literature review of the advances in MPC to handle hard constraints in both linear and nonlinear systems, with an emphasis on the results regarding stability and optimality. [118] summarized the previous 15 years of research in MPC and proposed several new directions for the future such as performance monitoring, process diagnostics, estimating states in nonlinear systems, improving system identification in the multi-input multi-output (MIMO) system context, and the challenges and reliability of the on-line dynamic optimization problem. Following these advances, MPC has been widely accepted in industry [77]. For example, in [79], MPC was implemented in an industrial automotive system and demonstrated superior closed-loop performance compared to traditional control schemes. However, as stated in [14, 15, 52, 116], a reliable process model that can capture the input-output relation of the dynamic system is essential to the success of advanced model-based control systems (e.g., Lyapunov-based MPC (LMPC) and economic MPC (EMPC)) as these systems require process models with well-characterized accuracy to predict the temporal evolution of the states, and thus, the identification of process models is a central pillar of control science and engineering.

Traditionally, mathematical and statistical models are widely used to derive process models in the field of chemical engineering. For instance, famous first-principles models such as the Navier-Stokes equations are crucial to modeling the fluid dynamics within physical systems. However, deriving first-principles models for chemical processes can be challenging due to the complexity of determining the fundamental physico-chemical phenomena of a process. In contrast, modeling dynamic systems using data-driven models has attracted significant attention, both historically and recently. In the context of developing dynamic models to embed into controllers, linear models have been studied exhaustively over the past few decades, leading to a well-developed framework and literature in this field. This is primarily due to the mathematical simplicity and results that can be derived for controllers with linear process models, despite the fact that most chemical processes exhibit highly nonlinear behavior and are characterized by numerous complex interactions between variables as seen in distillation columns [93] and catalytic continuous stirred-tank reactors (CSTR) [28]. For industrial process control systems, the parameters of a linear data-driven model are typically identified from industrial or simulation data [165]. A category of such data-driven models is autoregressive models such as autoregressive with exogenous inputs (ARX) or autoregressive–moving-average model with exogenous inputs (ARMAX) [107]. An ARX model takes the weighted sum of the lagged states and inputs to predict the current states and may also be reformulated as a linear time-invariant state-space model, which has been studied in-depth in the control literature. The primary reason linear models are still employed in the control of nonlinear processes is that, besides the aforementioned mathematical considerations, they are often able to be used for the design of a controller that can stabilize the nonlinear process itself. For example, [8] investigated the design of a Lyapunov-based EMPC (LEMPC) using empirical linear state-space models, derived conditions to guarantee closed-loop stability of the nonlinear process under the LEMPC based on the linear empirical model, applied the controller to a nonlinear chemical process, and found it to be a computationally efficient framework. Furthermore, when using a linear state-space model as the process model in MPC, its structure becomes similar to that of a linear-quadratic regulator (LQR), which has convex properties and guaranteed convergence [34]. In practice, this can lead to faster convergence of the MPC optimization problem compared to the usage of nonlinear models in MPC.

While the above methods perform well for linear systems, process modeling using linear models continues to be challenging for large-scale complex processes due to the limitation of enough and flexible parameters to capture all nonlinearities in the system. Even though linearized models can still provide accurate approximations around steady-state regions, as the process deviates from the steady-state regions, linear models' performance suffers because the linearization approximation no longer holds. In response, nonlinear autoregressive models such as nonlinear autoregressive with exogenous inputs (NARX) are constructed where a more flexible nonlinear mapping can be added to the lagged states and inputs to capture the nonlinear coupling between known input effects and unknown input effects [22, 74, 124]. One problem that autoregressive models have is the large number of parameters that need to be estimated. Therefore, extensive research has been done regarding methods to reduce the dimension of these models. For example, by using input projection methods such as principal component analysis (PCA) and partial least squares (PLS), the dimensions of autoregressive models can be reduced to a reasonable amount [128]. In addition, constraints to the number of parameters can be added to the model such as in the case of nonlinear additive auto regressive model with exogenous input (NAARX) [74]. However, even with the use of dimension reduction algorithms, it is still a laborious task, especially for multivariable systems, to balance between choosing a suitable model structure and identifying all the required parameters [91]. The field of nonlinear dynamic modeling remains an active area of research. In the recent literature, several methods have been proposed including explanatory approaches such as sparse identification for nonlinear dynamical systems (SINDy) [24], which directly identifies a nonlinear system as a first-order ordinary differential equation that may be integrated in time. For systems such as chemical processes that evolve on an attractor and reach a steady-state, explanatory methods such as SINDy have been shown to accurately capture the long-term trajectories of nonlinear systems in the presence of time-scale multiplicities [2] or high levels of sensor noise in the data [4]. The resulting SINDy models have also demonstrated closed-loop stability and faster convergence than first-principles models when incorporated into an MPC [3]. In [6], well-conditioned polynomial nonlinear state-space models were developed to be incorporated into an LEMPC. The model identification explicitly accounted for conditioning to yield well-conditioned models that could be integrated with a relatively large integration time-step, leading to significant reduction in computation time per sampling period. Other nonlinear dynamic modeling methods in the literature opt for black-box approaches such as Runge-Kutta time-steppers embedding neural networks to handle nonlinearities [58, 66, 130, 140] and entropic regression [11]. It is noted that these alternative methods exist and have their advantages as well as disadvantages, but a comparison study is beyond the scope of this paper.

Entering the era of big data, deep learning (DL) methods such as artificial neural networks (ANN) have gained much attention for their exceptional performance in capturing the behavior of complex physical systems, making them top candidates for model-based control systems. The large number of tunable parameters and the rich variety of nonlinear functions ANNs possess allow the capture of previously-considered “difficult nonlinearities”. When given the appropriate parameters, the universal approximation theorem states that an ANN is capable of capturing any complex input-output relation [42, 101]. However, ANNs did not gain popularity until recently due to the difficulties associated with implementation and training [128]. Specifically, for the modeling of complex physical systems, especially noisy and/or nonlinear processes, *deep* ANNs may be necessary to capture the input-output relation. Although training deep ANNs requires a large volume of data due to the high number of parameters to be estimated, which renders their implementation generally difficult, several recent developments have mitigated the challenges of training deep ANNs. Firstly, the proliferation of data generated by machines and devices in the last decade has made the training of data-demanding ANNs viable [183]. Secondly, while deep ANNs inherently involve many matrix operations, leading to large demands for computational power during training, recent advancements in computing infrastructure including cloud and parallel computing have made training deep ANNs feasible. Finally, the development of open-source machine learning libraries such as Tensorfow [1], PyTorch [123], and Keras have made the construction and training of complex ANNs much more straightforward and accessible.

There are many different types of ANNs; specifically, feed-forward neural networks (FNN) and recurrent neural networks (RNN) and their variants [35, 70, 143] have demonstrated potential for use in model-based control systems. Throughout the scientific history, FNNs have been proposed to be used as process models in MPC. [50] proposed the use of a two-hidden-layer FNN as the process model for dynamic matrix control (DMC), which is an early linear form of MPC. A more recent use of FNN is to model nonlinear dynamic processes since the multivariate nature of these processes leads to internal state interactions, causing difficulties for traditional models [5]. Therefore, research has been conducted to exploit the nonlinear nature of FNNs to develop process models for MPC. In [116], a dynamic FNN-based MPC was designed to control the interface level of a flotation column by manipulating the tailings flow rate. The dynamic FNN took in a multi-time step history of the states to predict the interface level after one-time step. The MPC's prediction horizon is set to contain multiple sampling periods, and each sampling period is equal to the prediction length of the dynamic FNN. Results showed that the dynamic FNN-based MPC performed better than traditional PID controllers due to faster convergence and smoother trajectories. [87] reported the use of a stacked FNN-based MPC for a multivariable nonlinear steel pickling process. FNNs were stacked in series, and an iterative method was used to obtain different future time steps of process states within the MPC's prediction horizon. The simulation results show that the FNN-based MPC displayed good convergence and stability even under disturbances and noise. In addition to using FNNs as the process models, FNNs can also be used as a model identifier to determine when a process is subject to parameter variations and uncertainties [73]. The identifier FNN updates the weights of the process model, which is a separate predictor FNN, in case of system variations and thus creates an adaptive neural network-based MPC. While the above use of FNNs as process models in MPC displayed good results, modeling complex long-term dynamic processes was not intuitive given the unidirectional structure of FNNs.

Therefore, the idea of incorporating recurrency to neural networks was proposed to better capture the ordinal nature within time-series datasets. The first RNNs can be traced back to the 1980s, when Hopfield networks were first created for pattern recognition purposes [78, 141]. There were also attempts to use RNNs to model nonlinear dynamic processes, but they were not widespread due to difficulties to train generalizable and accurate RNNs [114]. However, with recent progress in technology and neural network structure, RNNs have now emerged as a leading method to model nonlinear dynamic processes [143, 144]. [55] evaluated the applicability of several neural network architectures, including FNN and RNN variants, for different dynamic processes as surrogate models. For dynamic processes, FNNs may not perform as well as RNNs due to the lack of feedback connections that introduce past information derived from earlier inputs into the current output. Furthermore, the incorporation of feedback loops in RNNs leads to capturing dynamic behavior in a way conceptually similar to nonlinear dynamic models derived from first-principles [113]. As a result, modern RNN-based process models (e.g., Long short-term memory (LSTM) [76], Gated recurrent unit (GRU) [32], and encoder-decoder [155]) have been integrated with different model-based control schemes and used in many research fields, including chemical [193], mechanical [178], and pharmaceutical [168] engineering. Specifically for process control, [186] has conducted a comparative study on the use of LSTM- and GRUs as dynamic process models in predictive control for two chemical reactors. It was found that both models approximated the properties of the dynamic systems with high accuracy and drove the systems to desired set-points. For certain stochastic processes, a single RNN process model may not be adequate to capture the complex nature of the system; therefore, multiple RNNs can be used in conjunction, creating an ensemble in which the mean or median of the ensemble is taken as the final prediction. [175] utilized an ensemble of RNNs as the MPC process model to model the behavior of a fixed-bed catalytic reactor and demonstrated its performance with respect to computational fluid dynamics (CFD) results. In addition to traditional RNN architectures, an encoder-decoder architecture was proposed by [33] in which two RNNs were used in series to capture input sequences of various lengths and long-term dependencies. [189] have demonstrated that, for dynamic processes that contain many long-term dependencies, encoder-decoder-based RNNs perform better than LSTM/GRU-based RNNs. [96] have adopted an encoder-decoder RNN model to develop an MPC for the control of an HVAC system and demonstrated good convergence and stability. In addition to performance, [53] have argued that encoder-decoder models can easily be constructed from a model definition perspective for HVAC systems. Specifically, the list of inputs and outputs of encoder-decoder model is clearly aligned with the inputs and outputs of the HVAC process and this may help simplify model construction and reduce training costs of EMPC. Finally, [23] discusses the incorporation and evaluation of different RNN structures in MPC. Specifically, these RNNs were assessed from the perspective of a control system designer with emphasis on stability guarantees, safety verification, and consistency with the physical system for the RNN models. [173] proposed that the incorporation of physical knowledge into RNN models improves the performance of the MPC system. This claim was further supported by [9], in which two RNN models, one with physical knowledge and one without, were compared against each other for a two-CSTR-in-series system simulated on a high-fidelity chemical process simulator. It was found that the physics-based RNN-MPC system converged faster and required less computational time.

This article aims to survey the popular neural network modeling approaches and provide a tutorial on the construction and integration of these models with MPC. Recent advances in the development of neural network models for specific scenarios such as noisy data and on-line adaptation learning are presented as remarks throughout the tutorial. The explanations presented in this paper are meant to be accessible to a beginning graduate student with limited knowledge in control and machine learning. The remainder of the paper is organized as follows: in the next section, preliminary knowledge on the class of systems considered and stability assumptions are presented. The third section discusses the concept of MPC and real-time optimization (RTO) and their implementation in process control. In addition, the role of process model within MPC and RTO is presented with a focus on neural network model-based MPC and RTO. In the fourth section, the theories behind neural networks are discussed with an emphasis on the intuition behind their architecture. This survey on neural network models is not meant to be comprehensive, but to give readers an idea on the evolution of neural networks and background theories. The fifth section aims to give a brief tutorial on the construction of a neural network model. Starting from problem identification to model evaluation, the workflow is meant to be iterative and change with new findings during application. The sixth section gives an example of applying different neural network model-based MPCs and their performance on a chemical process. Finally, a summary and future directions of neural networks for MPC are discussed.

6.2 Preliminaries

6.2.1 Notation

The notation $|\cdot|$ is used to represent the Euclidean norm of a vector. A function $f(\cdot)$ is of class $\mathscr{C}^1$ if it is continuously differentiable in its domain. $\odot$ denotes the Hadamard product or element-wise multiplication. $:=$ denotes the assignment operator. The expression $x \to c$ represents the variable x approaching some constant c.

For time step notation, each time step represents a single integration step h_c, and Δ represents the sampling time, which is the time interval at which state measurements are available. $t = t_k$ is defined as the current time step, and any time before $t = t_k$ is considered historical information. M is the number of inputs that are fed to the model for prediction, which can include historical and present values of the state and manipulated input variables. N is the number of outputs that are produced by the model, which indicates the predicted process states in the next N time steps with an interval of h_c. The predicted outputs from the machine learning model constitute the intermediate and final states within and at the end of one sampling period Δ, respectively, where $\Delta = N \cdot h_c$. For each prediction, a sequence of historical and present information $(t_{k-M+1}, ..., t_k)$ is used to predict the future trajectory over a single sampling period $(t_{k+1}, ..., t_{k+N})$. Since the machine-learning model provides N future predictions over one Δ, the model will be called iteratively K times to solve the MPC with a prediction horizon of K sampling periods.

6.2.2 Class of Systems

In this work, the class of multi-input multi-output nonlinear continuous-time systems is considered and can be represented by the following state-space form:

$$\dot{x} = F(x,u) = f(x) + g(x)u \tag{6.1a}$$

$$y = h(x) \tag{6.1b}$$

where $x \in \mathbf{R}^n$ denotes the state vector, $u \in \mathbf{R}^m$ is the vector of manipulated inputs and $y \in \mathbf{R}^n$ represents the vector of state measurements which are available at every sampling period. The input vector is bounded by $u \in U = \{u_i^{\min} \leq u_i \leq u_i^{\max}, i = 1, ..., m\} \subset \mathbf{R}^m$. The terms $f(\cdot)$, $g(\cdot)$, and $h(\cdot)$ are sufficiently smooth vector and matrix functions of dimensions $n \times 1$, $n \times m$, and $n \times 1$, respectively, with the assumption that state and input variables are in deviation from their steady-state values such that the origin is a steady-state of the nominal system of Equation 6.1 (i.e., $(x^*_{ss}, u^*_{ss}) = (0,0)$, where the subscript "*ss*" indicates the steady-state). Throughout this tutorial review, initial time is assumed to be $t_0 = 0$, and all states are assumed to be measurable.

6.2.3 Stabilizability Assumption

The existence of a stabilizing feedback control law of the form $u = \Phi(x) \in U$ is assumed for stability considerations. The objective of this controller is to ensure that the origin of the nominal system of Equation 6.1 is exponentially stable under this controller. This assumption implies that there is a control Lyapunov function of class $\mathscr{C}^1$ denoted as $V(x)$ such that the following inequalities hold for all x within an open neighborhood D around the origin:

$$c_1|x|^2 \leq V(x) \leq c_2|x|^2, \tag{6.2a}$$

$$\frac{\partial V(x)}{\partial x} F(x, \Phi(x)) \leq -c_3|x|^2, \tag{6.2b}$$

$$\left| \frac{\partial V(x)}{\partial x} \right| \leq c_4|x| \tag{6.2c}$$

where c_1, c_2, c_3, c_4 are all positive real numbers. The controller $\Phi(x)$ can be constructed in the form of the universal Sontag control law proposed in [97]. Following [176], first, the open neighborhood around the origin, D, is identified where Equation 6.2b holds under the stabilizing controller $u = \Phi(x) \in U$. Then, the closed-loop stability region Ω_ρ is identified as a level set of $V(x)$ within the region D where $\Omega_\rho = \{x \in D \mid V(x) \leq \rho\}$, with $\rho > 0$. Moreover, the Lipschitz property of $F(x,u)$ with the upper and lower bounds on u implies the existence of positive constants Γ, $L_x, L_x^{'}$ such that the following inequalities hold for all x and $x' \in D$, and $u \in U$:

$$|F(x,u)| \leq \Gamma \tag{6.3a}$$

$$|F(x,u) - F(x',u)| \leq L_x|x - x'| \tag{6.3b}$$

$$\left| \frac{\partial V(x)}{\partial x} F(x,u) - \frac{\partial V(x')}{\partial x} F(x',u) \right| \leq L_x^{'}|x - x'| \tag{6.3c}$$

6.3 Real-time Optimization (RTO) and Model Predictive Control (MPC)

6.3.1 Real-time Optimization (RTO)

Real-time optimization (RTO) system is an advanced model-based process optimization and control tool to compute the optimum operating steady-state condition based on a user-defined objective function (e.g., minimum energy consumption, maximum economic profitability, etc. [192]). Typically, RTO is computed over a significantly longer window than the supervisory control layer. For example, RTO may be computed over hours or days while the supervisory control layer may be computed over minutes. As a result, the setpoint is updated in real-time according to the RTO output by addressing the optimization problem described in Equation 6.4:

$$\begin{aligned} \min_{x,u} \quad & L_e(x,u) \\ \text{s.t.} \quad F(x,u) &= 0 \\ g_p(x,u) &\leq 0 \\ g_e(x,u) &\leq 0 \end{aligned} \tag{6.4}$$

which minimizes a user-defined cost function expressed by $L_e(x,u)$. This function is commonly termed the economic cost function or economic stage cost, since it is a direct or indirect reflection of the process economics. The objective function can be designed to maximize typical chemical engineering performance measures such as the production rate of the desired product, selectivity of the desired product, and product yield. F is a steady-state model. As for g_p, it is a vector function, and represents process constraints (i.e., physical constrains) such as limits on inputs, and g_e is the economic constraints that includes oil and energy prices, etc. Both terms are matrix functions of dimensions $\mathbf{R}^p$ and $\mathbf{R}^e$, respectively.

6.3.2 Model Predictive Control (MPC)

MPC is an advanced control scheme that is able to anticipate the future states of the process to make intelligent decisions with respect to the constraints of the process. Theoretically, MPC consists of three main components (i.e., an objective function, a process model, and a real-time optimizer [27]), where the process model provides the prediction of state responses based on the underlying physical and chemical phenomena of the system. Subsequently, the real-time optimizer will compute the optimal control actions $u^*(t)$ for each sampling period (denoted by Δ) within the prediction horizon by solving a finite-horizon optimization problem with respect to the objective function and constraints. The operation of MPC is to address the following optimization problem:

$$\min_{u \in S(\Delta)} \int_{t_k}^{t_k+K\Delta} L(\hat{x}(\zeta), u(\zeta))\, d\zeta \tag{6.5a}$$

$$\text{s.t.} \quad \dot{\hat{x}} = F(\hat{x}(t), u(t)) \tag{6.5b}$$

$$\hat{x}(t_k) = x(t_k) \tag{6.5c}$$

$$u(t) \in U, \forall t \in [t_k, t_k + K\Delta) \tag{6.5d}$$

$$g(\hat{x}(t), u(t)) \in G, \forall t \in [t_k, t_k + K\Delta) \tag{6.5e}$$

where $\hat{x}$ represents the predicted states of the process model and $S(\Delta)$ denotes the set of piecewise constant functions with Δ. The optimal control actions are calculated to minimize the time integral of the objective function $L(\hat{x}(t), u(t))$ in Equation 6.5a over the prediction horizon K, meaning that the predicted trajectory over $K\Delta$ into the future is accounted for within the optimization problem. However, only the control action for the first sampling period will be implemented to the process system. In other words, K optimal input actions are computed for each Δ from the current time step $t = t_k$ until $t = t_k + K\Delta$ in a feedback control manner on the basis of the predicted state trajectory $\hat{x}$, but only the first optimal input $u^*(t_k)$ is applied to the process over the next sampling period and is held constant over Δ, which is known as sample-and-hold implementation. The predicted state trajectory $\hat{x}$ is computed using the process model of Equation 6.5b, which can be a first-principles or a data-driven model. Moreover, as shown in Equation 6.5c, the initial condition for the process model is obtained from the real-time measurement. In addition, the feasible range of control actions is defined in Equation 6.5d. Lastly, Equation 6.5e represents any additional constraints (e.g., Lyapunov stability constraints [176]) that are imposed to ensure that the optimal control actions meet the necessary conditions to guarantee closed-loop stability under sample-and-hold implementation.

Remark 6.1. *The MPC is implemented in a receding horizon manner so that, at every sampling time, the optimization problem is solved again when new feedback measurements are received. Note that state and input variables are typically represented in their deviation forms such that the steady-state values are at the origin. The operating steady-state values can be the optimal states and inputs that are calculated by the RTO, which is executed at a slower frequency compared to MPC.*

Remark 6.2. *The mathematical setup of MPC and RTO is similar to each other, but RTO is based on steady-state process models and MPC is based on dynamic models [127]. Therefore, both functionalities are critically dependent on the respective process models. As mentioned, the first-principles models are the primary candidates for this role. However, in practical operations, oftentimes an accurate first-principles model is not available. Thus, by replacing the process model with a neural network model, machine learning can be integrated with RTO and MPC to accomplish their respective goals using a data-driven modeling approach.*

6.3.3 Multi-layer Control Scheme

Traditional architecture-process optimization, particularly with regard to economic considerations, and chemical process control have been resolved in a hierarchical multi-layer control scheme, as illustrated in Fig. 6.1. Specifically, the first layer is the RTO, which solves for optimal set points in accordance with the steady-state process model and supervises the sublayers by sending out the optimization results. MPC is implemented in the second layer of this control

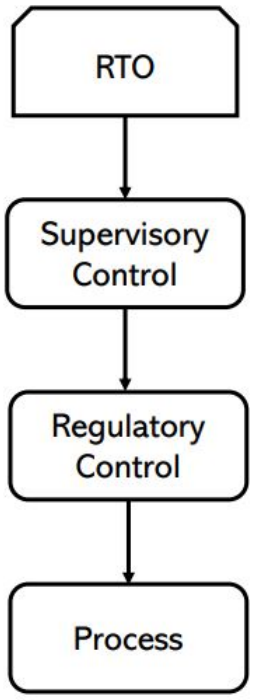


Figure 6.1: The typical process optimization and control perspective used in the field of chemical process industries.

scheme to direct and derive the process to the new operating point through manipulating the inputs with its constrained optimal control methodology, which takes into account physical limitations, process variable interactions, and predicted responses. Control actions from the MPC are employed to the system by the regulatory control layer.

In addition to the top two-layer control scheme, an EMPC, which introduces penalization terms into the objective function based on the economic performance of the process, has been proposed for its ability to achieve similar functionality without RTO and has attracted increasing attention recently [14,52]. However, regardless of the control scheme, a reliable process model is one of the most critical components of MPC and EMPC.

6.4 Neural Network Architecture Overview

6.4.1 Feed-forward Neural Networks (FNN)

Feed-forward neural networks are generally made up of multiple layers of neurons, with each neuron being a single logistic unit. Figures 6.2(a) and (b) show the schematics of an FNN and the model graph describing the calculations conducted within a logistic unit, respectively. The main concept behind the FNN or any neural network can be summarized into two parts: a forward pass to predict the desired output and a backward pass to update the weights and biases. In the forward pass, the input features are propagated through the neural network to generate the predicted output. In the backward pass, the predicted output is compared with the true output, and the error is back propagated through all the logistic units. The weights and biases of each logistic unit are updated with respect to the error to improve the next prediction.

Similar to how real neurons operate in the human brain, the forward propagation of a logistic unit functions analogously and is described mathematically by the following equations:

$$z_k^{[l]} = \sum_{j=1}^{n^{[l-1]}} h_j^{[l-1]} w_{jk}^{[l]} + b_k^{[l]} \tag{6.6a}$$

$$h_k^{[l]} = g^{[l]}(z_k^{[l]}) \tag{6.6b}$$

where superscripts in brackets, $[\cdot]$, denote the layer number of the neural network. l denotes the lth layer of the neural network that varies between 0 and L. Only generalized intermediate layer equations within a neural network are shown in Equation 6.6. The input layer is represented by $l = 0$ where $h_j^{[0]}$ is the equivalent of the input x_j and the output layer is represented by $l = L$ where $h_j^{[L]}$ is the equivalent of the predicted output $\hat{y}_j$. Subscripts j and k denote the jth and kth units within their respective layer in the neural network, b denotes the bias, and h denotes the hidden state. $w_{jk}^{[l]}$ is the weight connecting the jth neuron of the layer $l-1$ to the current kth neuron in the layer l. z denotes the weighted hidden states and the bias term. $n^{[l]}$ is the total number of neurons in the lth layer. In the forward propagation step, first, following Equation 6.6a, each logistic unit

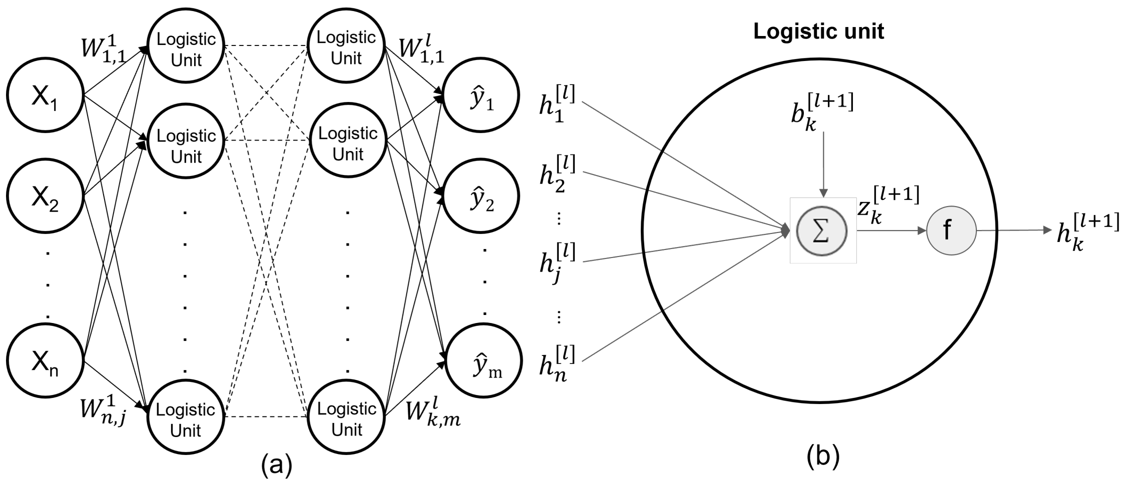


Figure 6.2: Schematics of (a) general FNN structure and (b) logisitic unit within FNNs.

takes inputs from all connected units from the previous layer, $h_j^{[l-1]}$, and aggregates the input signals together with its own bias, $b_k^{[l]}$. The summed signal, $z_k^{[l]}$, then undergoes a certain activation function, $g^{[l]}$, with common ones being the rectified linear unit (ReLu) or *tanh*, before being sent as the input $h_k^{[l]}$ to the next layer as per Equation 6.6b. During training and inference, signals start from the input layer and propagate to the final output layer L using this two-step procedure.

$$E = \frac{1}{R}\sum_{i=1}^{R} L(\hat{y}, y) \tag{6.7}$$

where E is the cost function, L is the loss function for each sample, i is the index of data samples, R is the total number of data samples, $\hat{y}$ is the predicted output, and y is the true output. During training, the cost function is minimized using optimization algorithms to find the optimal set of weights and biases.

Optimization algorithms are used to update the weights and biases within the FNN. Gradient descent (GD), when applied to weights and biases, respectively, is shown in Equation 6.8 below, and its variants are commonly used as the optimization algorithm for neural network training. The general structure of the GD algorithm is of the form,

$$w_{jk}^{[l]} := w_{jk}^{[l]} - \alpha \frac{\partial E}{\partial w_{jk}^{[l]}} \tag{6.8a}$$

$$b_k^{[l]} := b_k^{[l]} - \alpha \frac{\partial E}{\partial b_k^{[l]}} \tag{6.8b}$$

where $w_{jk}^{[l]}$ and $b_k^{[l]}$ are being updated and α is the learning rate and is conventionally positive (i.e., $\alpha > 0$). The main idea behind gradient descent is to find the local minimum of a differentiable function by iteratively moving in the opposite direction of the gradient of the function at each point. During neural network training, the gradient descent algorithm minimizes the loss at each iteration or epoch by varying the weights and biases. As shown in Equation 6.8, the gradient of the cost function with respect to each weight and bias needs to be calculated at each layer using the backpropagation algorithm. The backpropagation algorithm calculates the gradients starting from the final output layer and propagates to the input layer, which ensures computational efficiency by avoiding redundant calculations of intermediate terms [67]. Using the chain rule, the gradient can be transformed into two partial dervatives involving $z_k^{[l]}$ as shown below:

$$\frac{\partial E}{\partial w_{jk}^{[l]}} = \frac{1}{R}\sum_{i=1}^{R} \frac{\partial L_i}{\partial w_{jk}^{[l]}} = \frac{1}{R}\sum_{i=1}^{R} \left(\frac{\partial L_i}{\partial z_k^{[l]}} \frac{\partial z_k^{[l]}}{\partial w_{jk}^{[l]}} \right) \tag{6.9a}$$

$$\frac{\partial E}{\partial b_k^{[l]}} = \frac{1}{R}\sum_{i=1}^{R} \frac{\partial L_i}{\partial b_{jk}^{[l]}} = \frac{1}{R}\sum_{i=1}^{R} \left(\frac{\partial L_i}{\partial z_k^{[l]}} \frac{\partial z_k^{[l]}}{\partial b_k^{[l]}} \right) \tag{6.9b}$$

where L_i is the loss for the ith sample. For weights, in Eq. 6.9a, the partial derivative of $z_k^{[l]}$ with respect to $w_{jk}^{[l]}$ is the incoming hidden state of the previous layer, $z_j^{[l-1]}$. For biases, the partial derivative of $z_k^{[l]}$ with respect to $b_k^{[l]}$ is simply 1 due to the bias being a constant. Finally, as shown in Equation 6.9, the partial derivative of L_i with respect to $z_k^{[l]}$ is also needed to calculate the partial derivative of L_i with respect to w and b. This partial derivative is often denoted as the error term, $\delta_k^{[l]}$. Using the aforementioned simplifications, the gradient of the loss function with respect to weights and biases can be transformed into the following equations:

$$\delta_k^{[l]} = \frac{\partial L_i}{\partial z_k^{[l]}} \tag{6.10a}$$

$$\frac{\partial L_i}{\partial w_{jk}^{[l]}} = \delta_k^{[l]} z_j^{[l-1]} \tag{6.10b}$$

$$\frac{\partial L_i}{\partial b_k^{[l]}} = \delta_k^{[l]} \tag{6.10c}$$

where $\delta_k^{[l]}$ is the error of the kth neuron within the lth layer. The mathematical formulation of the error term is shown by the following equations:

$$\delta_k^{[L]} = (\hat{y} - y) g'(z_k^{[L]}) \tag{6.11a}$$

$$\delta_k^{[l]} = g'(z_k^{[l]}) \left(\sum_{r=1}^{n^{[l+1]}} \delta_r^{[l+1]} w_{kr}^{[l+1]} \right) \tag{6.11b}$$

where $g'(\cdot)$ is the derivative of the activation function $g(\cdot)$. $\delta_k^{[l]}$, as defined in Equation 6.10a, is calculated starting from the final output layer and propagated through to the input layer. At the output layer, assuming square loss, $\delta_k^{[L]}$ is equal to the difference between the true and predicted output multiplied by the derivative of the activation function, $g'(z_k^{[L]})$, as shown in Equation 6.11a. For all other layers, the calculation of $\delta_k^{[l]}$ requires the errors from the next layer $\delta_k^{[l+1]}$ and the derivative of the current activation function, $g'(z_k^{[l]})$ as shown in Equation 6.11b.

In this section, all equations are shown in vector form with individual neurons and weight connections labeled with subscripts j and k. In the following sections, all equations will be shown in their capitalized matrix form for simplicity. For example, Equations 6.6a and 6.6b are rewritten in their matrix form as follows:

$$Z^{[l]} = H^{[l-1]} W^{[l]} + b^{[l]} \tag{6.12a}$$

$$H^{[l]} = f^{[l]}(Z^{[l]}) \tag{6.12b}$$

where the dimensions of each variable in Equation 6.12 are as follows: $H^{[l-1]} \in \mathbf{R}^{n \times d}, H^{[l]} \in \mathbf{R}^{n \times h}, Z^{[l]} \in \mathbf{R}^{n \times h}, W^{[l]} \in \mathbf{R}^{d \times h}$, and $b^{[l]} \in \mathbf{R}^{1 \times h}$ where the superscripts of $\mathbf{R}$ represent the dimensions of the matrix. Specifically, n is the batch size, which is the number of samples that will be propagated through the neural network, d is the input size, and h is the number of hidden

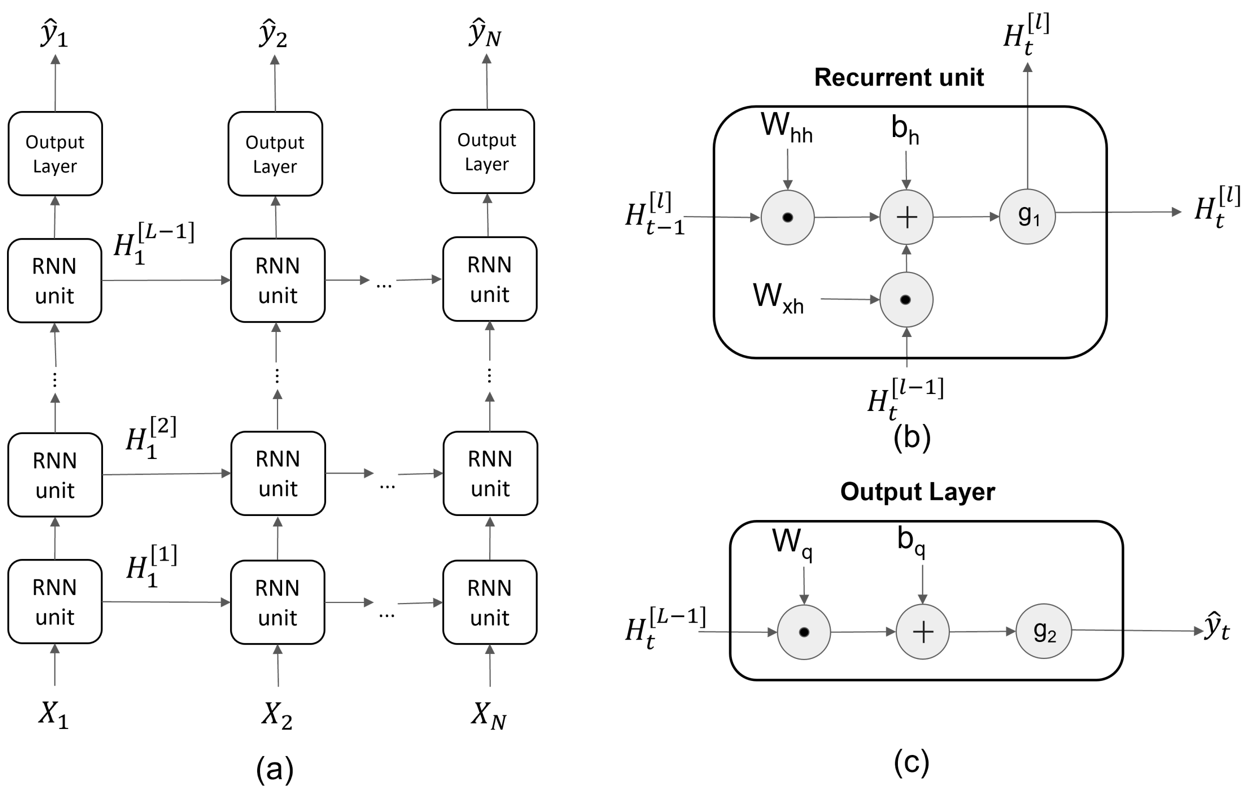


Figure 6.3: Schematics of (a) general unfolded RNN structure, (b) recurrent unit, and (c) output layer within RNNs.

units.

6.4.2 Sequential Neural Network Models

Sequential data is prevalent in many real-world problems and machine learning tasks—most popularly known for its role in natural language processing (NLP) and signal processing. Specifically, in chemical engineering, sequential data can exist in the form of sensor measurements. Sequential models are designed to account for the ordinal nature of these datasets. The main focus of this work will be on neural network model structures that are widely used to conduct time-series forecasting tasks for MPC.

6.4.2.1 Recurrent Neural Network (RNN)

RNNs can be thought of as FNNs with two dimensions instead of one, as shown by the unfolded diagram of an RNN in Figure 6.3(a). FNN inputs are batches of feature vectors $X_{FNN} \in \mathbf{R}^{n\times d}$, while RNN inputs are batches of sequential feature vectors $X_{RNN} \in \mathbf{R}^{n\times d\times t}$. Therefore, an additional dimension, t, is added to the neural network to account for the data's ordinal property. In FNNs, the output of each neuron, h, is propagated through the network to the last layer before making a final prediction. In multilayer RNNs, h is passed to both the next layer and the next ordinal input. H is a matrix that contains all the hidden states, h, within the same layer. Most RNNs are made up of two different types of units: a recurrent unit and an output unit. The recurrent unit inside the RNN takes in two inputs: the current input vector, X_t, and the previous hidden state, H_{t-1}. For simplicity, all following equations will omit the superscript l that refers to the layer number in the case of multilayer RNNs.

Forward propagation within RNNs is similar to that within FNNs and is described by the following equations:

$$H_t = g_1(X_t W_{xh} + H_{t-1} W_{hh} + b_h) \tag{6.13a}$$

$$O_t = g_2(H_t W_{hq} + b_q) \tag{6.13b}$$

where $X_t \in \mathbf{R}^{n\times d}, H_t, H_{t-1} \in \mathbf{R}^{n\times h}, W_{xh} \in \mathbf{R}^{d\times h}, W_{hh} \in \mathbf{R}^{h\times h}, b_h \in \mathbf{R}^{1\times h}, O_t \in \mathbf{R}^{n\times q}, W_{hq} \in \mathbf{R}^{h\times q}$, and $b_q \in \mathbf{R}^{1\times q}$ where q denotes the the dimension of the output. X_t represents the input matrix, H_t represents the hidden state, and O_t represents the output matrix where t denotes the current time step. W_{xh} is the input weight matrix, W_{hh} is the previous hidden state weight matrix, and W_{hq} is the output weight matrix. b_h and b_q are the bias terms associated with the hidden layer vector and the output vector, respectively. Figures 6.3(b) and (c) show a schematic of the operations conducted on the inputs within a recurrent and output layer, respectively. Within a recurrent layer, the calculation of H_t from one time step to the next is the same as shown in Equation 6.13a. Two different weight matrices, W_{xh} and W_{hh}, are generated for the current input and the previous hidden state. An optional bias vector, b_h, can also be included in the H_t calculation. The output unit is

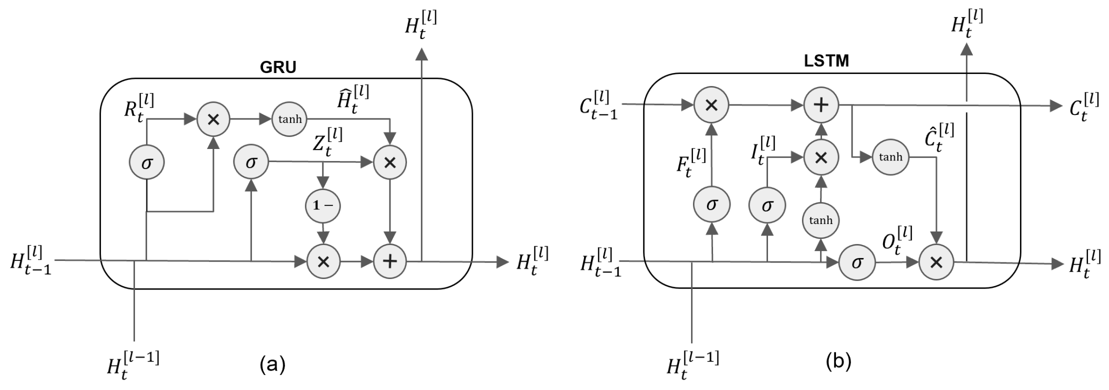


Figure 6.4: Schematics of (a) GRU and (b) LSTM.

analogous to a feed-forward logistic unit in that it contains a single weight matrix with a bias vector as shown in Equation 6.13b.

The backpropagation process within RNNs is much more complicated than that within FNNs because of the additional time dimension. In order to obtain the exact gradients with respect to all weights and parameters, the computational graph of the RNN needs to be calculated one time step at a time [164]. A problem with this calculation is that, when input sequences are long (¿1000), it could result in large matrix products, making the training computationally infeasible in many situations [188]. Another problem associated with long input sequences and gradients is divergent gradient values. As the input sequences become longer and the exponents of the weight matrices increase, the output becomes more likely to be divergent. This phenomenon is often referred to as vanishing or exploding gradients. A solution to this problem is to truncate the gradient at certain time steps to avoid large matrix calculations and divergent eigenvalues. However, truncation may result in loss of relevant information from the early time steps.

6.4.2.2 Gated Recurrent Unit (GRU)

Gating in neural networks refers to controlling the expression of key states in neurons. Generally, gating is achieved through the use of the gates which are sigmoid activation functions, $f(x) = \frac{1}{1+e^{-x}}$, that limit the output from 0 to 1. The output is then multiplied by the state variable to control its expression. As mentioned in the previous section, traditional RNN units have difficulties capturing longer-term dependencies due to the phenomenon of exploding or vanishing gradients. GRUs try to alleviate this issue by the addition of reset and update gates [36], and are mathematically described by the following equations:

$$R_t = \sigma(X_t W_{xr} + H_{t-1} W_{hr} + b_r) \tag{6.14a}$$

$$Z_t = \sigma(X_t W_{xz} + H_{t-1} W_{hz} + b_z) \tag{6.14b}$$

$$\hat{H}_t = \tanh(X_t W_{xh} + (R_t \odot H_{t-1}) W_{hh} + b_h) \tag{6.14c}$$

$$H_t = Z_t \odot H_{t-1} + (1 - Z_t) \odot \hat{H}_t \tag{6.14d}$$

where R_t, Z_t, and $\hat{H}_t$ denote the reset gate, update gate, and candidate hidden state, respectively. The subscripts r and z denotes the association of variables with the reset and update gate, respectively, and $R_t \in \mathbf{R}^{n\times h}, Z_t \in \mathbf{R}^{n\times h}, \hat{H}_t \in \mathbf{R}^{n\times h}, W_{xr}, W_{xz} \in \mathbf{R}^{d\times h}, W_{hr}, W_{hz} \in \mathbf{R}^{h\times h}$, and $b_r, b_z \in \mathbf{R}^{1\times h}$. As shown in Equation 6.14, the reset and update gates control the formulation of the candidate hidden state, $\hat{H}_t$, and the extent to which to update the hidden state, H_t, with the candidate hidden state. The reset and update gate values are calculated from the current input, X_t, and the previous hidden state, H_{t-1}, with the only difference being different multiplicative weight matrices and bias vectors.

The reset gate aims to capture short-term dependencies by controlling the extent of H_{t-1} that RNN should remember by limiting the expression of $\hat{H}_t$. Therefore, R_t is element-wise multiplied with H_{t-1} to control its expression in $\hat{H}_t$ as shown in Equation 6.14c. When $R_t \to 1$, $\hat{H}_t$ will be equal to the hidden state calculation of a traditional RNN unit. When $R_t \to 0$, $\hat{H}_t$ will be equal to the calculations performed within an FNN unit, since there is no information from the previous state.

The update gate aims to capture longer-term dependencies, as it controls the extent to which the new hidden state is a copy of the old hidden state by controlling the ratio between $\hat{H}_t$ and H_{t-1}. The current hidden state, H_t, is simply a weighted average between $\hat{H}_t$ and H_{t-1} controlled by Z_t, as shown in Equation 6.14d. When $Z_t \to 1$, H_t ignores $\hat{H}_t$ and subsequently, the current input X_t, resulting in $H_t = H_{t-1}$. In this case, the current hidden state is a direct copy of the previous hidden state. When $Z_t \to 0$, $H_t = \hat{H}_t$, but this does not imply that the previous hidden state is ignored. As seen in Equations 6.14c and 6.14d, H_t can still depend on H_{t-1} if $R_t \to 0$. Thus, only when both $Z_t, R_t \to 0$, will the current hidden state solely consider the current input and ignore all previous hidden states.

6.4.2.3 Long Short-term Memory (LSTM) Unit

GRUs attempt to solve the problem of vanishing and exploding gradients by directly changing the hidden state from one unit to the next. In contrast, the LSTM unit introduces a new state, the memory cell state C_t, to store additional information regarding short-term and long-term dependencies [76]. As shown by Figure 6.4(b), this new memory cell state is passed between LSTM units like a hidden state. In addition, the LSTM unit uses three different gates—the input, forget, and output gates—to dynamically update the values of previous hidden and memory cell states. The motivation behind the memory states and the three gates is similar to that of the GRU, which is to decide how to control the expression of previous states and current inputs in the hidden state.

Similarly to the gates in GRUs, the input, forget, and output gates are all activated by sigmoid functions using the current input, X_t, and the previous hidden state, H_{t-1}. Their mathematical formulations are shown below:

$$I_t = \sigma(X_t W_{xi} + H_{t-1} W_{hi} + b_i) \tag{6.15a}$$

$$F_t = \sigma(X_t W_{xf} + H_{t-1} W_{hf} + b_f) \tag{6.15b}$$

$$O_t = \sigma(X_t W_{xo} + H_{t-1} W_{ho} + b_o) \tag{6.15c}$$

where I_t, F_t, and O_t denote the input, forget, and output gate, respectively. The subscripts i, f and o denote the association of variables with the input, forget, and output gate, respectively, and

$I_t, F_t, O_t \in \mathbf{R}^{n \times h}, W_{xi}, W_{xf}, W_{xo} \in \mathbf{R}^{d \times h}, W_{hi}, W_{hf}, W_{ho} \in \mathbf{R}^{h \times h}$, and $b_i, b_f, b_o \in \mathbf{R}^{1 \times h}$. Specifically, the input, forget, and output gates will output values ranging from 0 to 1 to control the expression of key information in the new hidden state.

In an LSTM unit, instead of having a candidate hidden state as in a GRU, a candidate memory cell, $\hat{C}_t$, and memory cell state, C_t, are utilized to capture dependencies in the sequence and are shown by the following equations:

$$\hat{C}_t = \tanh(X_t W_{xc} + H_{t-1} W_{hc} + b_c) \tag{6.16a}$$

$$C_t = F_t \odot \hat{C}_{t-1} + I_t \odot \hat{C}_t \tag{6.16b}$$

where $\hat{C}_t, C_t \in \mathbf{R}^{n \times h}, W_{xc} \in \mathbf{R}^{d \times h}, W_{hc} \in \mathbf{R}^{h \times h}$, and $b_c \in \mathbf{R}^{1 \times h}$. The computation of $\hat{C}_t$ is similar to that of the three gates except with a *tanh* activation function that limits the output from -1 to 1 as shown in Equation 6.16a. I_t is used to introduce new information to C_t by controlling the expression of $\hat{C}_t$ in C_t. F_t is used to dictate how much old information is to be forgotten from the previous C_t through controlling the expression of C_{t-1} in C_t as shown in Equation 6.16b. Finally, as shown in Equation 6.17 below, the hidden state, H_t, is the element-wise product between O_t and $\tanh(C_t)$:

$$H_t = O_t \odot \tanh(C_t) \tag{6.17}$$

where O_t dictates how much of C_t is relevant to the current output and controls the extent of the contribution of C_t to H_t. A difference between LSTM and GRU is that even though the current input information may not be relevant to the current hidden state, this information is still stored in the memory cell state so that it can be used for the computation of future hidden states. In summary, by resolving the issue of vanishing and exploding gradients, the GRU or LSTM units will suppress irrelevant information and better capture longer-term dependencies using a combination of gates. For example, if an earlier input is highly significant for the prediction of future outputs, it is necessary to capture and store this dependency into the hidden state or a separate cell state. In addition, using a vanilla RNN unit without this storage may result in a very large or a very small gradient with respect to the weight matrices associated with the earlier inputs during training, and thus may cause exploding or vanishing gradient when propagated through the layers of the neural network.

6.4.2.4 Encoder-Decoder Architecture

One problem with traditional RNN architectures is that they struggle with variable-length input and output sequences [33]. In most popular NLP tasks such as language translation, the input and output sequence will likely have different lengths [155]. In the field of time-series forecasting, it can be beneficial to use previous sequences as opposed to a single point to predict a future sequence as certain patterns can be dependent on previous patterns. While the use of GRU and LSTM units alleviate this problem through the usage of memory cells, long-term dependencies are still difficult to capture due to the models' Markov property. Traditional RNN models rely on the previous state to fully capture even earlier states and do not have direct access to those early states. In the encoder-decoder system, the encoder has direct access to all past states within a certain historical window and thus can better capture long-term dependencies. In the context of MPC, the historical window used in calculating the prediction horizon should be tuned to capture different length dependencies within the time-series data. In order to address the aforementioned problems, the encoder-decoder system is designed with two major components: an encoder followed by a decoder. The encoder first takes a variable-length sequence as the input and summarizes it to a context state, which is passed to the decoder. The decoder then maps the context state to a variable-length sequence as the output. The encoder-decoder system can be thought of as a special RNN architecture, since each encoder-decoder unit can be any of the RNN/GRU/LSTM units shown in Figure 6.5.

The encoder takes a fixed input sequence with length $M-1$ and summarizes it into a context state that is passed to the decoder. In each encoder unit, each individual input x_i is transformed into the hidden state only and passed to the next encoder unit. As a result of this, the final context state, C, can be thought of as a function of all the previous hidden states within the historical window $C =$

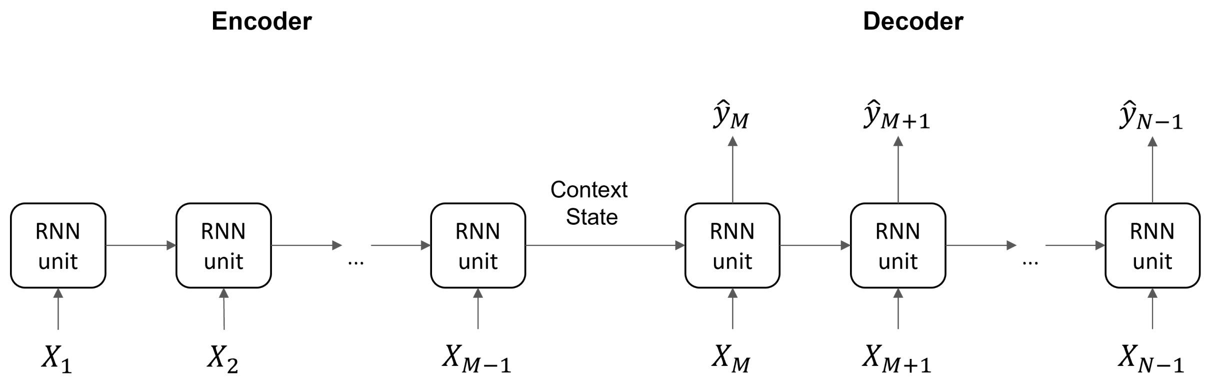


Figure 6.5: Structure of an encoder-decoder with RNN units.

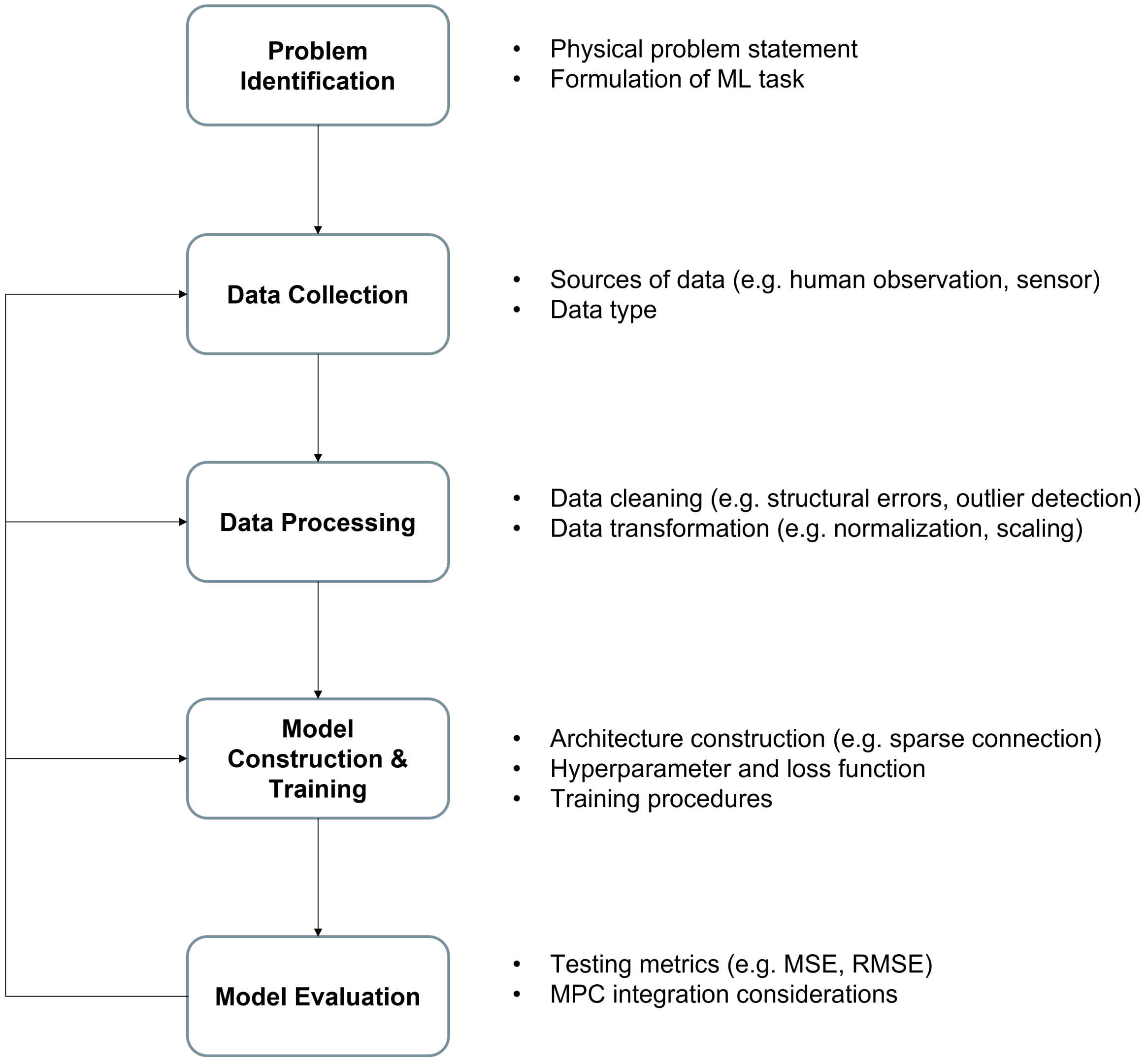
$f(h_{t_{k-M+1}}, ..., h_{t_k-1})$. The decoder uses the context state and decoder input sequence, $x_{t_k}, ..., x_{t_{k+N-1}}$, to predict the desired future sequence, $\hat{y}_{t_k}, ..., \hat{y}_{t_{k+N-1}}$. In the case of time-series forecasting, $\hat{y}_{t_k}$ is generally the predicted states for the next time step at $t = t_{k+1}$.

6.5 Neural Network Model Construction Tutorial

Incorporating machine learning to solve realistic problems is not a straightforward process. It is often a cyclical process that uses model evaluation to iterate between improving the data and improving the model. This cycle is crucial in developing a successful machine learning model, since it evaluates feedback from previous results and implements changes to further improve the earlier steps. A general workflow of the iterative cycle is outlined in Figure 6.6. In the following subsections, each step of the proposed workflow is explained in detail, with a focus on developing a neural network-based process model for MPC.

6.5.1 Problem Identification

The first step in developing any ML model is to identify a general problem statement and transform it into a specific ML task. For example, the problem statement might be to optimize the operation of a series of reactors. It is of utmost important to specify if the goal is to optimize a certain chemical species' yield, minimize environmental impact, or maximize overall profits. Even


Problem Identification
Physical problem statement
Formulation of ML task
Data Collection
Sources of data (e.g. human observation, sensor)
Data type
Data Processing
Data cleaning (e.g. structural errors, outlier detection)
Data transformation (e.g. normalization, scaling)
Model Construction & Training
Architecture construction (e.g. sparse connection)
Hyperparameter and loss function
Training procedures
Model Evaluation
Testing metrics (e.g. MSE, RMSE)
MPC integration considerations
Figure 6.6: Proposed neural network-based model construction workflow.

with a well-defined problem statement, there is still no clearly defined ML problem. Following the reactor example, if the goal is to optimize profit, an EMPC can be implemented to control the reactor in real-time to produce at the optimal rate under varying operating costs. For real-time control to be implemented, a dynamic model of the system must be constructed, which is where a neural network model may be considered. In the absence of a traditional first-principles model, a neural network surrogate model is able to provide accurate real-time state information of the reactor comparable to that of the traditional first-principles model [55]. At this step, the ML task details, such as classification versus regression and supervised versus unsupervised versus semi-supervised, should be formulated based on the task description and data availability. In supervised learning, ML algorithms are trained to learn the target relationship within datasets that contain both the inputs and the labeled outputs. In unsupervised learning, no outputs are given, and ML algorithms attempt to find possible relationships between inputs alone. Finally, in semi-supervised learning, unsupervised learning techniques are incorporated into supervised ML algorithms to avoid the need to label large amounts of data. In the case of most MPC and RTO problems, a regression neural network model trained with supervised learning is sufficient for implementation.

6.5.2 Data Collection

After the ML task is identified, it is important to understand the variety of data sources. This can range from physical devices to network traffic service information. In this work, some of the most common types of data sources will be highlighted to give the reader a general overview, focusing on physical equipment, such as sensors. Sensors are the core data sources for many chemical or manufacturing systems. They aim to measure explicit physical properties, such as temperature, pressure, and flow rate, at different levels of the system (device, subsystems, systems, and environment). Sensors are generally separated into two types: primary and secondary. Primary sensors measure the desired quantity directly, for example an infrared camera measuring temperature, while secondary sensors measure other quantities and calculate the desired quantity based on the measured quantity, for example a thermometer calculates the temperature by measuring the volumetric change of mercury. The major difference is the error and uncertainties produced by the two types of sensors. Secondary sensors are more prone to large uncertainties, as they may involve multiple measurements and thereby compound the error. In addition, secondary sensors are calibrated to the primary sensor, which ensures that the error is always equal to or greater than that of the primary sensor. Therefore, depending on the problem statement, levels of uncertainty and error need to be evaluated against the sensors' inherent error to see if the sensor's error range is acceptable. Regardless of how well the ML model is able to reproduce the given dataset, if the data error is too large, applications of the ML model will fail due to the discrepancy between on-line and off-line testing.

When implementing an MPC system, another point to consider is the sampling frequency at which the desired quantity can be measured or the control action can be enacted. In certain situations, the state can only be measured after large time intervals, resulting in a large sampling period with respect to the time constant of the process, which can cause the model to fail to effectively capture the process dynamics within each sampling period. In addition, due to the sample-and-hold implementation of the MPC, a large sampling period also implies that the MPC may be activated less frequently than the dynamics of the process, leading to performance deterioration. Therefore, the sampling frequency of states and the process behavior must be considered in the design of an MPC. An example of the above concern is the use of gas chromatography to measure the composition within a chemical reactor because the measurement and subsequent analysis may take more than ten minutes to report a concentration, which does not provide sufficient real-time concentrations for certain processes with fast dynamics.

Data collected from the system may come in a variety of different forms. Even when measuring the same physical property, the data can come in both structured and unstructured forms. For example, a sensor can produce time-series temperature data in the form of a numerical list or table, which is considered structured data. Alternatively, infrared cameras may generate a heat map that may be more relevant than time-series data but in an unstructured form. In addition, the data preprocessing step may vary as a result of the state of the process—static versus dynamic. In the chemical engineering context, static data can be thought of as steady-state, whereas dynamic data refers to transient behavior. For dynamic data, the order may be important to the process, whereas static data can be shuffled. Whether it is structured/unstructured or static/dynamic data, identifying the data type and quality is of utmost importance to ensure the success of the ML model.

6.5.3 Data Preprocessing

Data preprocessing is an essential step to convert the raw data collected from sensors and experiments to a clean and usable dataset (e.g., Modified National Institute of Standards and Technology (MNIST), UCI official dataset). The amount of work necessary to transform raw data to a usable dataset is often overlooked. Specifically, the first step is to remove duplicates and irrelevant observations from the raw dataset, which is almost inevitable in large datasets. In a manufacturing setting, multiple sensors are implemented but they may not all be relevant to the ML task. The main goal of data preprocessing is to reduce the size of the dataset while maintaining all the relevant information. The next step is to look for structural errors such as corrupted data values and missing or mislabeled features. The process of filling in missing data values is called data imputation, which can range from simply using the mean, median, or mode of the column to implementing simple machine learning techniques such as k-nearest neighbors. Interpolation methods can also be used depending on the nature of the dataset. Finally, after the dataset is treated for missing and irrelevant information, outlier detection methods can be implemented to increase the performance of the ML models. Common outlier detection methods include Z-score, probabilistic models, and clustering methods. For parametric datasets, which are datasets with a known distribution, the Z-score method proves to be an efficient way to eliminate outliers and is shown below:

$$z = \frac{x - \mu}{\sigma} \tag{6.18}$$

where x is the current data point, μ and σ are the mean and standard deviation within the dataset, and z is the Z-score for the data point. As shown in Equation 6.18, Z-score uses the mean, μ, and the standard deviation, σ, to assess whether a data point is an outlier or not. The common Z-score threshold is around $\pm 2.5 \sim 3.5$. Clustering methods can be used to detect outliers for non-parametric datesets, and specifically, the density-based spatial clustering of applications with noise (DBSCAN) method is particularly effective. DBSCAN focuses on finding neighbors by density on an "n-dimensional sphere" with radius, ε [56]. DBSCAN aims to cluster all the outliers to an out-of-bound cluster where they can be further processed or removed.

In addition to off-line outlier detection methods, on-line outlier detection algorithms have been proposed to deal with dynamic time-series data. [98] have proposed an algorithm that combines an autoregressive moving average process model with a modified Kalman filter that uses past and current data to estimate the current data point and its variation. [85] have proposed the use of deep neural networks in detecting outliers within time-series data. Specifically, the raw time-series data is enriched with more statistical features and an autoencoder is used to select the most representative statistical features. Outliers often have non-representative features and thus deviations from the enriched time-series data is taken as outliers. Finally, [94] have used generative adversarial networks (GAN) for anomaly detection in time-series data. The proposed GAN framework uses LSTM units as the basis for its discriminator and generator to capture the temporal correlation in the time-series dataset. The discriminator itself is a direct tool for anomaly detection. The generator is exploited to capture the mapping from the latent space to the real data distribution, and this distribution can be used to detect anomalies in the test dataset. A combination of the discriminator and generator aspect of the proposed GAN is used as a metric to classify outliers within time-series data.

For any ML project, it is important to have at least two sets of data for training and testing purposes. In the case of neural network models, a third validation split is also recommended for model tuning. A typical split ratio is 80/20 or 70/15/15 with validation, although these ratios can and should be tuned based on the problem statement and availability of data. The training dataset is used to adjust the weights and biases of the neural network, the validation dataset is used to adjust the hyperparameters (number of neurons, number of layers, etc.) of the neural network, and the testing dataset is used to evaluate the performance of the neural network to an unseen dataset. Data splitting is conducted before any further data processing to prevent data leakage. Data leakage refers to the leakage of information, such as the mean or standard deviation of the testing to the training dataset, that can affect the testing accuracy. The next step in data preparation is data processing, which refers to the application of different transformations to the dataset to improve training performance. Data scaling, which applies some type of scaler to normalize the dataset within a certain range, can be applied to both structured and unstructured data, making it commonly the first transformation applied to a dataset. This prevents large discrepancies in the gradient between different input features during model training, which can cause weight values to change dramatically, making the training process unstable. The two most commonly used scalers are the Min-Max scaler and the Z-score scaler. A min-max scaler, shown in Equation 6.19 below, scales the entire dataset's values between the user-defined feature range and is calculated using the following equation:

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}(f_{max} - f_{min}) + f_{min} \tag{6.19}$$

where x is the current data point, x_{max} and x_{min} are the maximum and minimum values within the dataset, f_{max} and f_{min} are the user-defined maximum and minimum feature values, and x_{scaled} is the min-max scaled data point. Generally, f_{max} and f_{min} are 1 and 0 by default, respectively, which results in x_{scaled} to be within the range of 0 to 1. On the other hand, the Z-score scaler, shown in Equation 6.18, scales the dataset to a zero-mean distribution with unit standard deviation. After data scaling, more task-specific transformations can be implemented to generate the input tensor. Based on the type of ML library (Tensorflow, Keras, Pytorch) and the type of neural network chosen (FNN, RNN), the input shape of the training data is processed differently. Custom helper functions can be implemented to transform the original dataset into the desired training shape.

Remark 6.3. *For time-series forecasting tasks, data cannot be entered into the model for training* *as a single sequence. For the training of neural networks, the input sequence needs to be partitioned into fixed length intervals of historical and prediction sequences. In other words, the neural network training sequence would have a total length of $M+N$ with the input being the historical window plus the instantaneous measurement and the output being the prediction sequence of length N. A common algorithm to achieve this sequence format is the sliding window algorithm [45]. In the sliding window algorithm, criteria for the desired window are defined, and the window is slid across the entire sequence with a fixed step size. In this scenario, the criteria are that the total window length is equal to $M+N$ and less than the total sequence length.*

6.5.4 Model Construction

Depending on the ML task, there are various models that can be applied. As mentioned previously, the focus of this article will be on supervised regression models for dynamic data with consideration for control purposes. A variety of different models are compared with clearly defined trade-offs in performance, interoperability, and computational cost.

6.5.4.1 Model Architecture

Feed-forward Neural Networks (FNN)

Several different neural network structures were introduced in Section 4, and all of them can be applied to modeling nonlinear dynamic systems depending on the specific process and computational resources available. When constructing the process model for MPC, it should be noted that the MPC does not require the entire prediction sequence within one sampling period from the process model; as a minimum, only the last time step output, $t = t_{k+N}$, is required, where $t_{k+N} - t_k$ corresponds to one sampling period Δ in the MPC formulation. Therefore, the simplest case of a process model is a MIMO FNN model with the input being the process states and manipulated input at the current time step, $t = t_k$, and the output being the process states at the end of the prediction sequence, $t = t_{k+N}$. Note that this time step is analogous to the integration time step of numerical integration methods when simulating a system of ODEs. A variation of this formulation is the addition of the previous time steps as input to the FNN as shown in [116]. With this formulation, the FNN will take in M different time steps of process states and manipulated inputs to predict the process states at $t = t_{k+N}$. As proposed in [87], another method is to train an FNN that only predicts a single time step ahead; this FNN can be called repeatedly for N times, or N such FNNs can be trained and stacked together, to obtain the process states at time step $t = t_{k+N}$. This iterative method is similar to the calculations done in RNNs but without the feedback/recurrent connections within individual logistic units. It is important to note that it is beneficial to use a framework that is able to predict intermediate states rather than only the last states within each sampling period. This is because intermediate states provide more information on the predicted state trajectory and provides a better representation of the cost function of the MPC optimization problem, since the cost function is typically in the form of an integral over the entire prediction horizon. Having access to intermediate states with a shorter time interval in between also allows for better numerical approximations that are necessary within the MPC optimization.

Recurrent Neural Networks (RNN)

While directly implementing a MIMO FNN can satisfy the information needed by the MPC to calculate the optimal control trajectory, this approach may not result in the best performance due to the loss of intermediate information. Therefore, RNNs can be used in place of FNNs as capturing the ordinal aspect of dynamic dataset improves the performance since information between the first and last prediction time steps may have a significant impact on each other [115]. Based on the iterative FNN model, an improvement is to replace the feed-forward logistic units with recurrent units. Specifically, recurrent units will be used to account for the N future time steps at which the process states will be predicted i.e., every intermediate time step between $t = t_k$ and $t = t_{k+N}$. In addition, different types of recurrent units, such as LSTM or GRU, can be used in place of normal recurrent units to improve performance for certain processes. [186] have compared the performance between GRU and LSTM as process models in MPC for a chemical reactor system. Both performed better than regular recurrent units, and GRU was recommended due to its lower computational cost. It is important to note that the choice between recurrent units is highly process and data dependent.

Other than using different types of neural network, the specific connections within a neural network can also be customized to improve performance, interoperability [23], and computational efficiency [180]. [173] have developed a physics-based RNN model with knowledge of the structure of the process. Specifically, for two CSTRs in series, two partially-connected LSTM layers were used to model the input and output connections of each CSTR. Compared to the fully-connected LSTM model, the partially-connected model allowed for the decoupling between the second CSTR's input from the first CSTR's process states, which was also reflected in the real system. From a control perspective, the decoupling effect simplifies the system by reducing the interactions between the control actions resulting in faster convergence and better stability. [9] have performed a comparative study on RNNs that incorporate physical knowledge and traditional fully-connected RNNs in the context of a large-scale complex chemical process simulated with Aspen Plus Dynamics. The physics-based RNNs displayed superior accuracy and computational efficiency compared to the fully connected RNNs, which resulted in better convergence speed and stability in MPC integration.

Encoder-decoder Neural Networks

For certain dynamic processes, such as HVAC systems, long-term dependencies and past sequential trends are particularly important to predicting future process states. Therefore, a special architecture of RNNs, encoder-decoder RNNs, can be used to address the above problems. Compared to the input of traditional RNN structures, the input of the encoder-decoder system will be the sequence from $t = t_{k-M+1}$ to $t = t_k$ rather than the current time step $t = t_k$. The encoder then summarizes all the historical information within time period, $t = t_{k-M+1}$ to $t = t_{k-1}$, into a single context state, which is used in the decoder as the initial hidden state. Additionally, the decoder takes input at the current time step, $t = t_k$, and additional inputs at future time steps, $t = t_{k+1}$ to $t = t_{k+N-1}$, if they are available and relevant to the prediction process. The implementation of the encoder-decoder system may vary depending on the ML library used, but a general pseudocode structure is shown in Pseudocode 1. In Pseudocode 1, by defining *return_sequence* = *True*, the RNN unit will return an output at every time step rather than only once at the end of the sequence. Similarly, by defining *return_state* = *True*, the RNN unit will return all states in addition to the final hidden state. In the case of a LSTM unit, the cell state can also be accessed through the *return_state* argument. [189] have demonstrated that for air pollution data that contain many long-term dependencies, encoder-decoder-based RNNs perform better than LSTM/GRU-based RNNs. One thing to note is that encoder-decoder RNNs typically use more computational resources during both training and inference, and for most dynamic processes, an LSTM or GRU is sufficient to capture the system behavior. Therefore, it is important to start with a simple baseline model and build up the complexity of the neural network models. Finally, as there always exists a trade-off between model performance and computational efficiency, it is recommended to clearly track each model's accuracy and complexity after training.

Remark 6.4. *Instead of constructing a single neural network model, multiple models can be created from the same dataset and used together to predict the future process evolution. This is also known as ensemble learning and is known to have several advantages over using a single neural network model. First, ensemble learning allows for more generalization and prevents overfitting as the algorithm can be exposed to different subsets of the dataset through n-fold cross-validation methods. In other words, ensemble learning can reduce the variance of the algorithm while maintaining a low bias for individual models [126]. Second, due to the nonlinearity of the neural network models, the optimization associated with the process model is non-convex and is an NP-hard problem. Therefore, through the use of different weight initialization methods, the neural network model can potentially avoid getting trapped at local minima and arrive at more optimal sets of weights that allows the neural network to accurately approximate the latent function that transforms input sequences to corresponding output sequences [175]. Third, uncertainty during model selection can be better accounted for using ensemble learning algorithms than individual learning algorithm [112].*

There are multiple ways to construct an ensemble learning algorithm. In particular, two general categories, homogeneous and heterogeneous, will be discussed in this paper.

Pseudocode 1: Encoder-decoder Architecture

```
Encoder:
    Input
    {
        layer class: input layer
        shape: (Batch size) × (Encoder Length) × (Number of States and Inputs)
    }
    (input: encoder_input_data)
    output: encoder_input
    Recurrent Layers (can be repeated)
    {
        layer class: LSTM or GRU
        hidden units: n_encoder
        return sequence: False
        return state: True
    }
    (input: encoder_input)
    output: encoder_state_h, encoder_state_c

Decoder:
    Input
    {
        layer class: input layer
        shape: (Batch size) × (Decoder Length) × (Number of States and Inputs)
    }
    (input: decoder_input_data)
    output: decoder_input
    Recurrent Layers (can be repeated)
    {
        layer class: LSTM or GRU
        hidden units: n_decoder
        return sequence: True
        return state: False
    }
    (input: decoder_input, initial state: encoder_state_h, encoder_state_c)
    output: decoder_sequence
    Dense Layers (can be repeated)
    {
        layer class: Dense
        hidden units: n_decoder
    }
    (input: decoder_sequence)
    output: prediction_sequence
```

Homogeneous ensemble learning algorithm consists of models with a single base learning algorithm in which individual models are trained from different subsets of the given dataset through methods such as n-fold cross validation and bootstrap sampling. In n-fold cross validation, n different training and testing dataset splits are conducted and each dataset split is used to train a different model [48]. In bootstrap sampling, also commonly known as bagging, several bootstrap datasets take samples from the initial training set without replacement. As a result, multiple bootstrap training sets are constructed in which the initial training set may appear once, more than once, or may not appear at all [47]. Individual learning models are then trained using the different datasets and a weighted average is used to integrate the individual models together. Conversely, heterogeneous ensemble learning refers to the compilation of different learning algorithms from the same dataset. Specifically, multiple machine learning methods, such as FNN, RNN, support vector regressors (SVR), and gradient boosting machines (GBM) can promote better diversity within the ensemble learning algorithm and thus improve performance [136, 187].

6.5.4.2 Hyperparameter Tuning

Another important aspect of model development is the tuning of model hyperparameters. In machine learning, hyperparameters refer to parameters that are defined by the user rather than those optimized during training. These can include the number of neurons and layers, the optimizer, the mini-batch size, and others.

The number of layers and neurons is often very important to the model performance as it defines the size and complexity of the model. Depending on the complexity of the input-output relationship, the size of the neural network should be constructed appropriately. While increasing the number of layers and neurons will increase the training performance, it will also make the model less generalizable, which is also known as overfitting. On the contrary, using an overly simplified model to model a complex process leads to not capturing input-output relationships, which is a form of underfitting. Methods to recognize overfitting and underfitting will be explained more in detail in the model training section.

In addition to changing the size and complexity of the neural network, the choice of optimizer is also very important for the final performance of the model. Three popular optimizers, stochastic gradient descent (SGD) with momentum, RMSprop, and *Adam*, are shown in Equations 6.20a, 6.20c, and 6.20e, respectively, below:

$$w := w - \alpha m \tag{6.20a}$$

$$m = \beta m + (1-\beta)\nabla E \tag{6.20b}$$

$$w := w - \frac{\alpha}{\sqrt{v+\varepsilon}}\nabla E \tag{6.20c}$$

$$v = \gamma v + (1-\gamma)\nabla^2 E \tag{6.20d}$$

$$w := w - \frac{\alpha}{\sqrt{v+\varepsilon}} m \tag{6.20e}$$

where w, m, and v are the weight, the momentum of the gradient, and the moving average of squared gradients, respectively, and are updated at every epoch. β and γ are hyperparameters for momentum and adaptive learning rate, respectively, ε is a sufficiently small constant (10^{-8}), and E is the cost function. In Section 4.1, the most fundamental and core ML optimization strategy, gradient descent, is shown in Equation 6.8. GD updates the weights and biases after every epoch which means the full gradient is calculated for all observations at each epoch. Although accurate, calculation of the full gradient for every epoch is slow and can cause long computation times for large datasets. SGD offers a solution for this problem by approximating the full gradient through calculating the gradient in mini-batches. Subsequently, a momentum term m, as shown by Equation 6.20b, can be added to the SGD algorithm to accelerate the rate in which gradients move, leading to even faster convergence as shown in Equation 6.20a. Up to this point, when the weights are updated, the learning rate, α, is constant for all weights. However, the magnitude of the gradient can be different for different weights, which leads to the creation of an adaptive learning rate. Thus, RMSProp, shown in Equation 6.20c, adds a square root term of the square of the gradient v, where v is calculated in Equation 6.20d. Finally, the aforementioned two methods, adaptive learning rate and momentum, can be combined into one which is the *Adam* optimizer, as shown in Equation 6.20e. [86] has demonstrated that the *Adam* outperforms other stochastic optimization methods and works well on practical datasets in most situations.

Remark 6.5. *The mini-batch size refers to how many observations are used when calculating the gradient and updating the weights. Learning rate is another important hyperparameter. Learning rates that are too small lead to slow convergence of the model, but larger learning rates may lead to missing the optimal solution. Therefore, adaptive learning rate algorithms, such as* Adam*, are highly recommended.*

Several searching algorithms can be applied to conduct hyperparameter tuning. The most intuitive method is grid search in which only one hyperparameter is varied at a time to evaluate the change in model performance with respect to the varied hyperparameter. While grid search is comprehensive, the computational cost of grid search is immense when associated with a high dimensional hyperparameter space. Therefore, random search was developed to reduce the large computational cost of grid search through randomly selecting combinations of hyperparameters instead of enumerating all possible combinations [19]. Random search greatly outperforms grid search when only a small amount of hyperparameters affect the model performance. In addition to random search, Bayesian-based and gradient-based hyperparameter optimization methods have also been introduced [104, 151]. Manually implementing a hyperparameter search algorithm is labor and skill intensive. Therefore, commerical services, such as Google's HyperTune, are candidate options for conducting hyperparameter tuning.

6.5.4.3 Cost Functions and Regularization

In Section 4, a generic cost function is shown in Equation 6.7. The cost function is the mean of all errors within a dataset while the loss function refers to the error of a single datapoint. For regression tasks, the absolute error and the square error are good choices for the loss function within the cost function. For certain logarithmic datasets, the square logarithmic error can also be used if the data points cannot be scaled during the data preparation step. Since the cost function considers the entire dataset, the mean of the loss of all the individual data points is taken as the cost function, leading to the terminology of mean absolute error (MAE), mean square error (MSE), and mean square logarithmic error (MSLE). Each of the aforementioned loss functions can be transformed into their respective cost function using the following equations:

$$E_{MAE} = \frac{1}{R}\sum_{i=1}^{R} |y_i - \hat{y}_i| \tag{6.21a}$$

$$E_{MSE} = \frac{1}{R}\sum_{i=1}^{R} (y_i - \hat{y}_i)^2 \tag{6.21b}$$

$$E_{MSLE} = \frac{1}{R}\sum_{i=1}^{R} (\log(y_i + 1) - \log(\hat{y}_i + 1))^2 \tag{6.21c}$$

In addition to selecting a loss function, a regularization term can be added to the cost function for smoothing purposes and to prevent overfitting. Three of the most common regularization methods, L_1, L_2, and Elastic net, are shown below:

$$E_{L_1} = \frac{1}{R}\sum_{i=1}^{R} L(y,\hat{y}) + \lambda \sum_{j=1}^{n} |W_j| \tag{6.22a}$$

$$E_{L_2} = \frac{1}{R}\sum_{i=1}^{R} L(y,\hat{y}) + \lambda \sum_{j=1}^{n} W_j^2 \tag{6.22b}$$

$$E_{L_{elastic}} = \frac{1}{R}\sum_{i=1}^{R} L(y,\hat{y}) + \lambda \left(\theta \sum_{j=1}^{n} |W_j| + \frac{1-\theta}{2} \sum_{j=1}^{n} W_j^2 \right) \tag{6.22c}$$

where λ is the regularization hyperparameter and θ is the Elastic net hyperparameter. Both L_1 and L_2 regularization methods are based on the concept of using different L_p norms to penalize high regression coefficients for complex models to avoid overfitting. L_1 regularization, or Lasso regression, penalizes the absolute value, L_1 norm, of the weights as shown in Equation 6.22a. L_1 regularization is generally used for sparse feature sets, since it can perform feature selection by zeroing out irrelevant features' weights. L_2 regularization, or Ridge regression, penalizes the square, L_2 norm, of the weights as shown in Equation 6.22b. L_2 regularization cannot eliminate features due to the nature of the L_2 norm, but performs better than L_1 regularization in most cases. Elastic net, shown in Equation 6.22c, linearly combines L_1 and L_2 regularization through a weighted sum and adds an additional hyperparameter, θ, to adjust the ratio between the two methods. In the aforementioned regularization methods, a regularization parameter λ is included to control the extent of regularization within models. Elastic net outperforms both regularization methods in most situations, especially when the number of features is much larger than the number of observations [195].

6.5.5 Model Training

During model training, it is important to save the training history of the model. The training history can indicate whether the model has experienced trends of over- or under-fitting. The most common plot shows the evolution of the loss function with respect to epochs during training and validation. In most cases, longer training will result in model overfitting to the training dataset and will not generalize well to the validation and testing datasets. In the case of overfitting, the training loss keeps on decreasing while the validation loss increases rather than decreases as more epochs pass. An early stopping function can be implemented to prevent excessive training through the monitoring of validation loss. For example, an early stopping criteria of a certain number of consecutive increasing validation loss can be defined and the training process will stop if the criteria is reached. On the contrary, in underfitting, both the training and validation loss is very high and is still decreasing at an exponential rate. Ideally, a good model should have similar training and validation loss at the end of training. A good application to keep track of all training data is Tensorboard which automatically plots all training and validation curves as well as model structure graphs.

Remark 6.6. *In realistic processes, noise is inevitably present due to a combination of process disturbances and sensor errors. In the earlier data preparation step, outlier detection techniques, such as DBSCAN, are introduced to eliminate anomalies. However, a majority of the noise is within operating range and inherent to the data points. Many different training routines are developed* *to combat noisy datasets. The dropout method is developed to prevent overfitting in large neural networks through randomly dropping connections to neurons during training [152]. Dropout has then been extended to RNN models and adapted with Monte Carlo method to handle noisy data during training [61, 171]. In addition to the dropout method, [71] has developed a new training method called "Co-teaching" to effectively train robust deep neural networks. The "Co-teaching" method trains two neural networks in parallel and allows the exchange of information during each mini-batch. During forward propagation, each neural network selects a subset of assumed "clean" data and sends them to the other network. During backpropagation, each neural network updates its weights using only the "clean" data sent by the other network. [171] has applied both of these techniques to an example of a chemical process with the integration of MPC. It is demonstrated that both methods display superior performance compared to the baseline model in fitting and MPC set-point convergence.*

Remark 6.7. *Up to this point, only neural networks trained from a fixed dataset have been discussed. In reality, a process may change over time due to a mixture of external (e.g., equipment degradation and disturbances) and internal (e.g., fouling within equipment) factors. Therefore, in the presence of model uncertainty and parameter variations, on-line adaptive learning is essential to maintaining an accurate and up-to-date process model. At the same time, the frequency with which process models are retrained needs to be limited by event-triggered schemes in order to improve applicability and efficiency of adaptive control systems [156, 160]. In [73], a neural network identifier has been trained to detect process variations and update the parameters of the process model. [7] has proposed an adaptive EMPC system through the use of an error-triggered model re-identification scheme in which a threshold is set between the predicted and measured states as a trigger for model update. Building on top of the previous works, [172] has developed a dual event and error-triggered on-line update scheme for RNN models in a Lyapounov-based MPC. Specifically, the event-triggered model re-identification occurs when a triggering condition based on state measurements is violated, while the error-triggered update scheme is activated when the accumulated RNN modeling error exceeds some error threshold. With the proposed framework,* *the adaptive RNN-based LMPC performs better than a standard RNN-based LMPC in terms of guaranteed stability and control action smoothness.*

6.5.6 Model Evaluation

After the machine learning model has been constructed and trained, it is necessary to evaluate the trained model with meaningful metrics. In this section, general machine learning metrics are discussed as well as specific errors for MPC. Previously, the training and validation errors were used to check the training process and tune the model parameters. In model evaluation, the testing dataset is used to evaluate the finalized model's performance in terms of both accuracy and generalization on a set of unseen data. In the context of process modeling, the testing dataset may contain operating conditions unseen in the training operating conditions. For time-series forecasting tasks, general regression testing metrics that can be used include MAE, MSE, mean average percentage error (MAPE), and root mean square error (RMSE). These error metrics describe how close the model's predicted states are to the true process' states. Hence, errors are used to check the training process and tune the model parameters. Specifically, it is necessary to ensure that the training error is below a certain bounded modeling error threshold in order to guarantee exponential stability for the nominal system of Equation 6.1 under a Lyapunov-based controller built using an RNN model. Subsequently, the generalization of the model needs to be tested using a set of unseen data. In the context of a specific process, the testing dataset may contain some unseen operating conditions. For time-series forecasting tasks, general regression testing metrics can be used that include MAE, MSE, mean average percentage error (MAPE), and root mean square error (RMSE). These error metrics describe how well the model predicts the future process states. It has been demonstrated through simulations that lower testing metrics lead to improved stability in the control system. However, there are also other aspects that should be considered in the context of integrating control. For example, the smoothness of the prediction trajectory is very important for the stability the control action implemented.

Remark 6.8. *A more generalized error bound for RNN can be calculated using statistical machine* *learning theory. It is a measure of how well a neural network hypothesis learned from training data generalizes to unseen data so that it is more comprehensive for a wide range of operating conditions rather than the given training and testing conditions [169, 174]. The generalization error bound is found to be such that it does not have any dependency on the states of the neural network, as it is dependent on the weights, sample size, length of the input sequence, and width and depth of the neural network.*

6.6 Neural Network-based MPC Implementation: Chemical Process Example

In this section, a nonlinear chemical process is used to demonstrate the performance of various LMPCs. In particular, three different neural network models (FNN, RNN, and encoder-decoder) are considered as the process models for LMPCs. The specific chemical reactor example has been chosen to be tractable in terms of the understanding of its dynamic behavior and evolution by chemical engineers and researchers familiar with chemical processes.

6.6.1 CSTR Process Description

Consider a non-isothermal, well-mixed CSTR, where the following reversible first-order exothermic reaction is taking place:

$$A \leftrightarrow B$$

The following mass and energy balance equations represent the first-principles model that describes the process dynamics:

$$\frac{dC_A}{dt} = \frac{1}{\tau}(C_{A0} - C_A) - r_A + r_B \tag{6.23a}$$

$$\frac{dC_B}{dt} = \frac{-1}{\tau}C_B + r_A - r_B \tag{6.23b}$$

$$\frac{dT}{dt} = \frac{1}{\tau}(T_0 - T) + \frac{-\Delta H}{\rho C_p}(r_A - r_B) + \frac{Q}{\rho C_p v} \tag{6.23c}$$

$$r_A = k_A e^{\frac{-E_A}{RT}} C_A \tag{6.23d}$$

$$r_B = k_B e^{\frac{-E_B}{RT}} C_B \tag{6.23e}$$

where the notations C_A and C_B represent the concentration of chemical A and B, respectively. The reactor temperature is denoted as T, the inlet temperature is denoted by T_0, and C_{A0} is the inlet concentration. Also, for the reaction kinetics, the reaction rate constant and the activation energy for the forward reaction are denoted as k_A and E_A, respectively, and k_B and E_B for the backward reaction. The residence time of the reactor is τ, the heat capacity of the liquid mixture is denoted by C_p, the volume of the reactor is denoted by v, and the reaction enthalpy is ΔH. The CSTR is surrounded by a heating/cooling jacket that provides/removes heat at a rate Q to/from the reactor. The optimal steady state point for the process described in Equations 6.23. Table 6.1 lists the values of the process parameters along with the steady-state values at the optimal operating point.

Table 6.1: Parameter and steady-state values for the CSTR.

$C_{A_{ss}} = 0.4977\ mol/L$	$\tau = 60\ s$
$C_{B_{ss}} = 0.5023\ mol/L$	$Q_{ss} = 40386\ cal/s$
$C_{A0_{ss}} = 1\ mol/L$	$V = 100\ L$
$T_o = 400\ K$	$T_s = 426.743\ K$
$k_A = 5000\ /s$	$E_A = 1 \times 10^4\ cal/mol$
$k_B = 10^6\ /s$	$E_B = 1.5 \times 10^4\ cal/mol$
$R = 1.987\ cal/(mol\ K)$	$\Delta H = -5000\ cal/mol$
$\rho = 1\ kg/L$	$C_p = 1000\ cal/(kg\ K)$

The control objective in this example is to drive the process states, C_A, C_B, and T, to the optimal steady-state point by manipulating the heating rate Q and the inlet concentration C_{A0}. The process variables are all considered in deviation form from their steady-state values, which gives $x^T = [x_1,\, x_2\,, x_3] = [C_A - C_{A_{ss}}\,, C_B - C_{B_{ss}}\,, T - T_{ss}]$ such that the origin is the equilibrium point of this system. Furthermore, this extends to the manipulated inputs $u = [u_1, u_2] = [Q - Q_{ss}\,, C_{A0} - C_{A0_{ss}}]$, where the control action u is bounded by a lower bound $u^{LB} = [-40{,}000\ cal/s,\ -1\ mol/L]$ and an upper bound $u^{UB} = [40{,}000\ cal/s,\ 2\ mol/L]$.

The nonlinear minimization problem of LMPC is resolved using the interior point optimizer (IPOPT) package for each sampling time, which is $\Delta = 10\ sec$ with $h_c = 0.5\ sec$. IPOPT is an open source optimization package that can be employed to solve nonlinear optimization problems [20]. It employs an interior point-line search filter method, which tries to find a local optimum. Furthermore, the objective function of the LMPC can be formed as $L(x,u) = x^T Ax + u^T Bu$, where A and B are diagonal penalty matrices. Matrices A and B are critical for the MPC performance and are tuned according to the guidelines discussed in [10]. The chosen Lyapunov functions are $V(x) = x^T Px$, where $P = diag\{10^5, 10^5, 1\}$ is a positive definite matrix.

6.6.2 Data Generation and Processing

In this work, extensive open-loop simulations is conducted using the first-principles model to build the training and testing dataset. Specifically, starting from various initial conditions, the process model is integrated using the explicit Euler method under time-varying manipulated inputs while recording the states' evolution at each time step (i.e., $x_{t_{k+1}}, \ldots, x_{t_{k+N}}$, where $N = \Delta/h_c$). Subsequently, the dynamic time-series data are transformed into the format required for training and testing through the use of the sliding window algorithm. In our case, the criteria for the sliding window is that the total window length is equal to the sum of the input window length, M, and prediction horizon length, N. The open-loop simulation runs for 50 time steps for 432 different initial conditions, and the first 30 time steps are used as the training dataset, while the remaining 20 time steps are used as the testing dataset. In total, 43200 data samples were available for training and testing. The goal is to identify whether the model can learn the behavior of the process within the first 30 time steps to predict the next 20 time steps starting from different initial conditions. Both the training and testing datasets are scaled with respect to themselves only to avoid information leakage. The training and testing datasets are scaled from 0 to 1 using Equation 6.19 to avoid large discrepancies between gradients during training. Finally, the training and testing datasets are reshaped into 3-D tensors with input and output dimensions of $(R_{train}, M+N, n_{states}+n_{inputs})$ and $(R_{test}, M+N, n_{states})$, respectively, where R refers to the total number of training/testing observations and n_{states} and n_{inputs} refers to the number of states and inputs, respectively.

6.6.3 Neural Network Models Construction

In this work, the performance of using three different types of neural network models: FNN, RNN with LSTM units, and encoder-decoder with LSTM units is investigated. A general hyperparameter search is conducted on the hidden layers, hidden units, and activation functions for each of the neural network models. The final FNN is constructed with a single fully-connected layer with 10 hidden units and a linear activation function. The RNN consists of a single LSTM layer with eight hidden units and one fully-connected output layer on top with N hidden units. The LSTM layer and fully-connected layer have activation functions of *tanh* and linear, respectively. The encoder-decoder model consists of two LSTM layers (one encoder and one decoder layer) connected at the ends with 8 hidden units each and a single fully-connected output layer on top of the decoder LSTM layer with N hidden units. The LSTM layers and fully-connected layer have activation functions of exponential linear unit (ELU) and linear, respectively. Due to the different model architectures, the inputs to the models are slightly different. In FNN, the time step t_k will be used to predict the time step t_{k+N}. In RNN, the time step t_k will be used to predict the future sequence $t_{k+1},...,t_{k+N}$. In the encoder-decoder system, the historical and present information $t_{k-M+1},...,t_k$ will be used to predict future time steps $t_{k+1},...,t_{k+N}$. All these models are comparable since, in our MPC implementation, only the last time step, t_{k+N}, is used in the objective function to determine the optimal control trajectory for one Δ. The neural network model will be called K times to determine the control trajectory for the full MPC prediction horizon.

For training, an MSE cost function is used, and each model is trained for 100 epochs with a callback function that would follow an early stopping criterion. Each model's training and validation errors are logged to ensure no overfitting or underfitting occurs. From the training and validation errors, the FNN model is rejected because its MSE is two orders of magnitude larger than the other two models. Therefore, the FNN is not tested in both open and closed-loop simulations to save computational resources due to its poor training performance.

Remark 6.9. *The architecture and training procedures of the three models are kept simple because of the good quality of the simulation dataset and the large number of available operating conditions. In real scenarios, data may be contaminated with noise or be insufficient in quantity, and thus different architectures or processing steps need to be explored to combat these problems. Some methods for combating noisy data are included in Section 5.4.*

6.6.4 Open-loop performance

Subsequently, after the development of different NN models, an open-loop simulation was performed to evaluate the generalization of the models and their ability to capture the dynamics of the given CSTR process. During open-loop simulation, the two manipulated inputs, the heating rate Q and the initial concentration C_{A0}, were varied to compare the states predicted by the models versus the ground truth given by the first-principles model. Figure 6.7 illustrates the open-loop prediction using the LSTM model and the encoder-decoder model in response to time-varying inputs. Three different combinations of $[Q - Q_{ss}, C_{A0} - C_{A0_{ss}}]$ with values of $[1.5 \times 10^4, 0], [3 \times 10^4, 1.5]$, and $[2 \times 10^4, 1]$ were introduced in the forms of two step changes at $t = 100$ sec and $t = 200$ sec. The two models' trajectories from the plot are in good agreement with the first-principles model. Specifically, the MSE for each state given by the LSTM and encoder-decoder models in the open-loop simulation is computed and shown in Table 6.2. Since the MSE values are sufficiently small for both models, it can be concluded that the two models provided decent accuracy to move to the closed-loop simulation.

Table 6.2: MSE comparison of the open-loop prediction results between the neural network models and the first-principles model.

	$C_A - C_{A_{ss}}$ [mol/L]	$C_B - C_{B_{ss}}$ [mol/L]	$T - T_{ss}$ [K]
LSTM	3.24×10^{-6}	1.00×10^{-6}	3.61×10^{-2}
Encoder-decoder	5.29×10^{-6}	7.84×10^{-8}	1.69×10^{-2}

6.6.5 Closed-loop Performance

After the open-loop simulation, the models were tested in a closed-loop simulation of the underlying process under LMPCs based on the LSTM model and the encoder-decoder model. The dynamics of the closed-loop process under each controller is illustrated in Figures 6.8 and 6.9. The states trajectories are shown in Figure 6.8, while the associated manipulated inputs (i.e., control actions) are shown in Figure 6.9. From the resulting trajectories, it can be seen that the LMPC based on the encoder-decoder model outperforms the RNN-based LMPC in terms of smoothness of the state profiles, but the controller was able to drive the three states towards steady-state and stabilize the system with both models. Furthermore, the encoder-decoder based LMPC was able to drive its states closer to the steady-state state values as shown in Table 6.3.

Table 6.3: Final offset for each state from steady-state under LMPC using RNN and encoder-decoder models.

	$C_A - C_{A_{ss}}$ [mol/L]	$C_B - C_{B_{ss}}$ [mol/L]	$T - T_{ss}$ [K]
LSTM	3.7×10^{-3}	1.8×10^{-2}	7.3×10^{-2}
Encoder-decoder	7.8×10^{-5}	-1.2×10^{-4}	7.9×10^{-3}

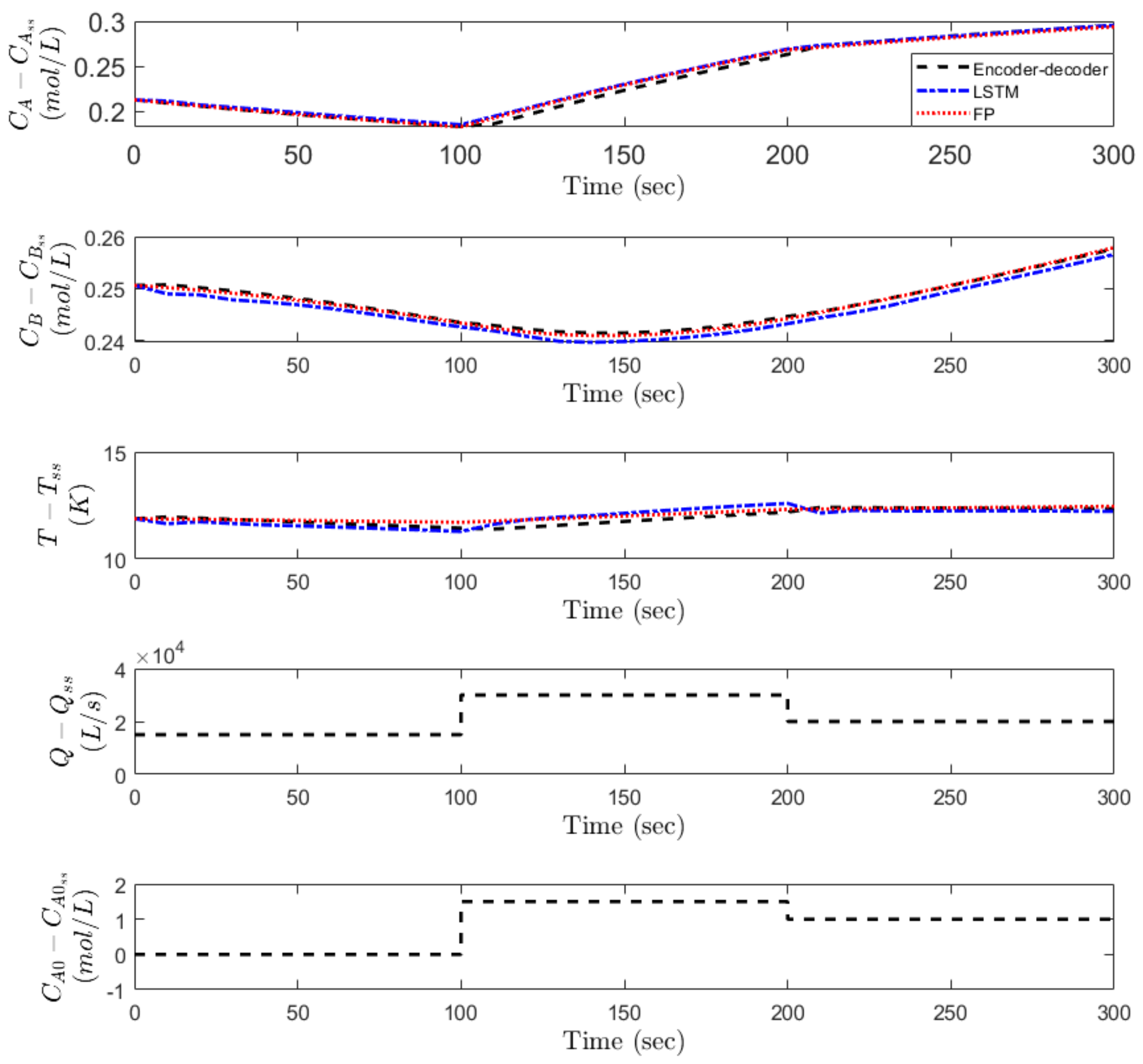


Figure 6.7: Open-loop state and manipulated input profiles for the CSTR example.

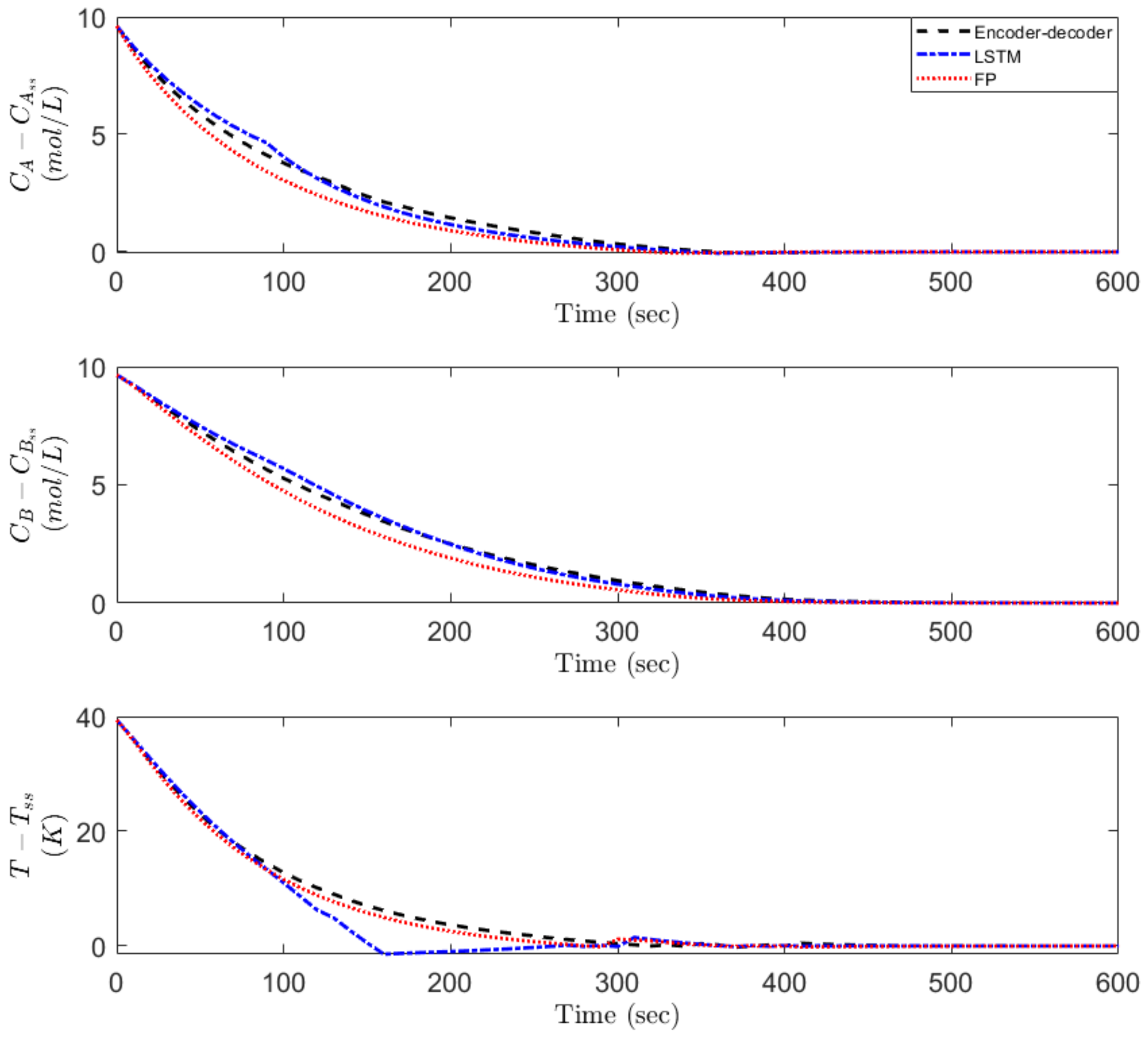


Figure 6.8: State profiles of the closed-loop simulation of the first-principles process model under the LMPC using three models: first-principles (FP), LSTM, and encoder-decoder.

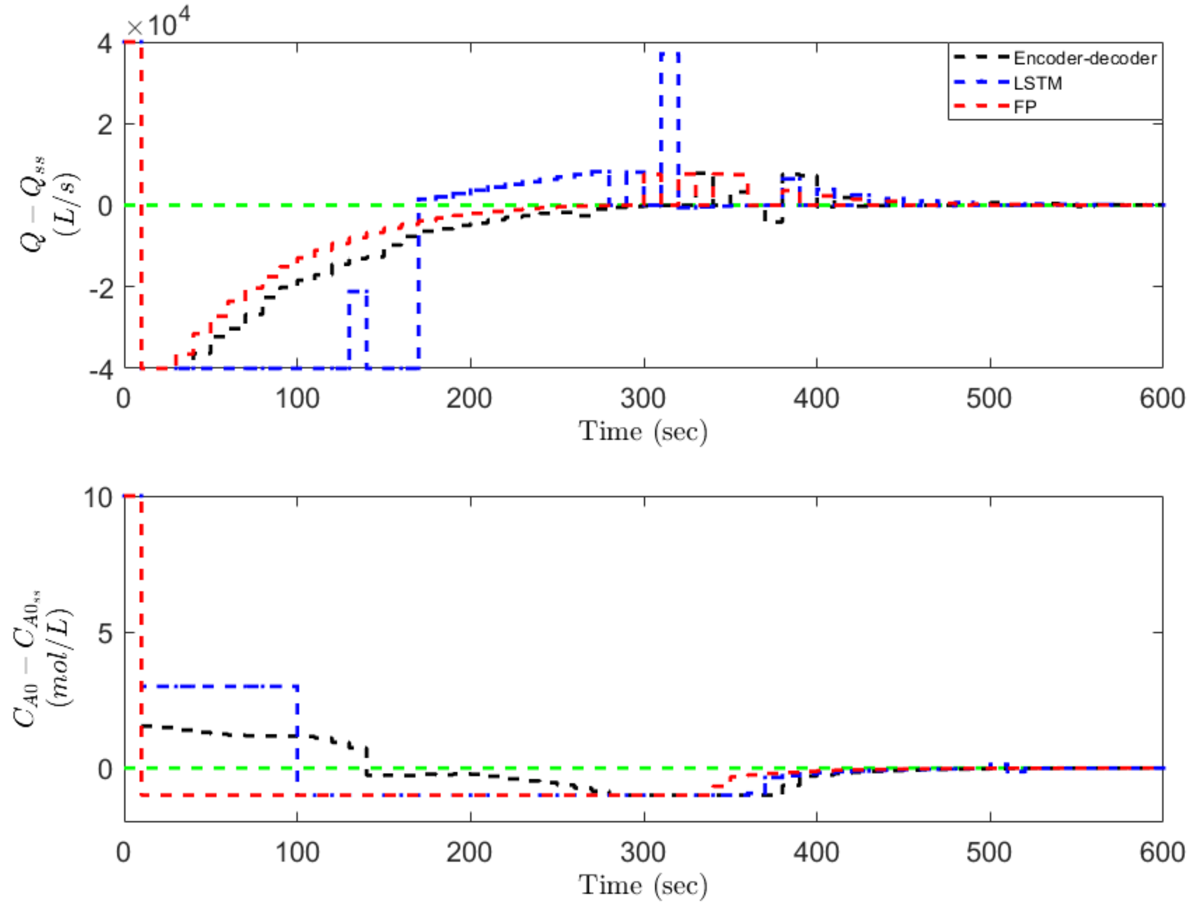


Figure 6.9: Input profiles of the closed-loop simulation of the first-principles process model under the LMPC using three models: first-principles (FP), LSTM, and encoder-decoder.

6.6.6 MPC Computation Time Considerations

One area not explored in the neural network-MPC example is the computational time necessary to find the optimal control action. In practice, there is a limit on the time allocated for the MPC to spend on solving the nonlinear optimization problem to ensure closed-loop stability. Some factors that affect the MPC's speed are the model's inference time, initial guesses, and optimization solver. Depending on what type of model is used in the MPC, the model's inference time can vary dramatically. A simple first-principles model takes less time than a traditional machine learning model, which takes less time than a deep neural network learning model. The inference time between different neural networks can also differ depending on their size and architecture. The inference time of the models used in this work is in the range of $0.1 \sim 0.5$ seconds. In addition, the initial guess and the solver choice used in the optimization problem can also greatly affect the computational load. If the MPC optimization problem is to be solved with a poor initial guess, an inadequate solver, or if the predictive model takes too long to calculate, it is possible that the nonlinear optimization solver will not converge to a solution in the time allotted, resulting in a suboptimal MPC [147]. The bottleneck in the MPC is heavily process-dependent and needs to be identified when attempting to improve the controller performance. For example, in a very complex process, the model required to capture the dynamics may be architecturally advanced and lead to a high model inference time, while a high-dimensional system will require a large number of values to be provided as initial guesses, which may be non-trivial and the bottleneck in solving the MPC optimization problem.

6.7 Conclusion and Future Directions

A survey on several neural network modeling approaches, in particular FNN, vanilla RNN, GRU, LSTM, and encoder-decoder architecture-based RNN, and their integration with MPC was discussed in this work. In addition, a tutorial was provided on the construction of the aforementioned neural network models with remarks on dealing with specific scenarios such as noisy data. Finally, a chemical process example was studied in closed-loop under the different neural network model-based MPCs to demonstrate the advantages and disadvantages of each model.

For future research directions, a recently proposed family of neural networks, named neural ordinary differential equations (ODE), provides the potential to improve the performance of modeling of continuous time-series data. This method proposes to parameterize an ODE between the states of a neural network and, as a result, the output of the neural network is the solution of an ODE initial value problem, which is computed with an explicit ODE solver (e.g., Euler method, Runge–Kutta methods) [31]. Compared to traditional RNNs that are usually interpreted as discrete approximations of time-series data, theoretically, neural ODEs can be interpreted as continuous approximations of the data. In particular, recent applications of neural ODEs in the literature include improved performance for forecasting time-series data, especially for data sets with large or irregular sampling times [139]. A potential future direction includes using neural ODEs as the process model for MPC to improve the performance of the MPC when dense and synchronous measurements are not available for model training purposes.

On the other hand, another neural network approach to consider is the transformer architecture. Up to this point, only sequential neural network models such as RNN, LSTM, GRU, and encoder-decoder systems were discussed. These sequential models generally have a recurrent structure which allows them to easily capture the ordinal aspect of time-series data. However, a drawback of the recurrent property is that they are not optimized for parallel computation. The input sequence to sequential models is provided the inputs one element at a time, which does not allow for parallel training and batch inference. In the example of an RNN, the hidden state of one RNN unit needs to be calculated first before being used as an input to the next RNN unit. In addition, sequential models assume that the previous recurrent unit is able to fully capture past behaviors as the current unit does not have direct access to past non-immediate units. Therefore, [159] have developed a new deep learning architecture, called transformer, in order to overcome these limitations associated with sequential models. In the context of MPC, the implementation of transformers can potentially not only improve model accuracy but also speed up the MPC's computation time due to the faster model inference as a result of parallelization.

Chapter 7

Conclusion

This dissertation provides a number of applications of machine learning methods, particularly neural networks, to additive manufacturing focusing on process monitoring and data analytic tasks. Specifically, Chapter 2-4 focused on the development of ML-assisted process monitoring tools for the DMLS process. Chapter 5 proposes a smart manufacturing data transfer framework between the machine, factory, and cloud levels.

Chapter 2 presented an integrated cross-validation framework between different in-situ sensors to combine the strength of the sensors and lower individual biases. First, the DMLS process was simulated using the FEM to obtain the real-time dynamic heat transfer behavior. The FEM model data was then processed in two different ways to replicate both the MPM and OT sensor data. Subsequently, automatic process monitoring tools were developed using CNNs that were trained from the simulated data. Finally, a cross-validation algorithm-based on CNN classification statistical analysis was developed to lower individual sensor bias toward certain errors. Chapter 3 expanded on the FEM model developed in Chapter 2 by constructing multi-scale models of the process to increase fidelity and decrease computational load. Specifically, a micro- and meso-scale model were developed to simplify the laser source and powder-level heat transfer behavior so that a part-scale model was able to be developed. The part-scale model result was further processed to mimic experimental sensor results and used in conjunction with experimental sensor data to develop a process monitoring CNN. Finally, the maintenance of the CNN was explored with a focus on data transfer and storage.

In Chapter 4, an application of machine learning methods to non-image sensor data was discussed. In particular, the recoater arm interaction during layer-to-layer print was correlated with the powder level thickness, which was predicted using a feed-forward neural network. Compared to the traditional polynomial regression method of predicting powder level thickness, the feed-forward neural network was able to detect low powder thickness much better and achieved a narrower 95% confidence interval. Finally, experiments were conducted to validate the proposed method for detecting the interaction of the recoater arm during the printing process.

In Chapter 5, a data-driven process optimization and data transfer framework dedicated to additive manufacturing processes were proposed. The framework outlined three levels of hierarchy: the machine level, the factory level, and the cloud level, where each level has its own duties and limitations. The aforementioned machine learning and FEM modeling methods were also integrated into the proposed framework to incorporate Industry 4.0 concepts to further automate the manufacturing process. Furthermore, due to the large data size typically involved with AM processes, a data-transmission scheme was also proposed to serve as a data transmission guideline and a case study with the appropriate data scale for a typical AM application was presented. Finally, the connection and similarities between the proposed AM framework and the Industry 4.0 framework was elaborated to provide further motivation for the research in this field.

In addition to AM, Chapter 6 reviewed the role of popular neural networks, in particular FNN, vanilla RNN, GRU, LSTM, and encoder-decoder architecture-based RNN, in MPC was conducted focusing on their intuition and applicability. In addition, a tutorial was provided on the construction of neural network models with remarks on dealing with specific scenarios such as noisy data. A CSTR chemical system example was also studied in a closed-loop under the different neural network model-based MPCs to demonstrate their performance in terms of convergence and stability. Finally, future research directions on using neural ordinary differential equations and transformers in MPC was proposed.

Bibliography

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, pages 265–283, Savannah, GA, USA, 2016.

[2] Fahim Abdullah, Zhe Wu, and Panagiotis D Christofides. Data-based reduced-order modeling of nonlinear two-time-scale processes. *Chemical Engineering Research and Design*, 166:1–9, 2021.

[3] Fahim Abdullah, Zhe Wu, and Panagiotis D Christofides. Sparse-identification-based model predictive control of nonlinear two-time-scale processes. *Computers & Chemical Engineering*, 153:107411, 2021.

[4] Fahim Abdullah, Zhe Wu, and Panagiotis D Christofides. Handling noisy data in sparse model identification using subsampling and co-teaching. *Computers & Chemical Engineering*, 157:107628, 2022.

[5] Abdul Afram, Farrokh Janabi-Sharifi, Alan S Fung, and Kaamran Raahemifar. Artificial neural network (ANN) based model predictive control (MPC) and optimization of HVAC systems: A state of the art review and case study of a residential HVAC system. *Energy and Buildings*, 141:96–113, 2017.

[6] Anas Alanqar, Helen Durand, and Panagiotis D Christofides. On identification of well-conditioned nonlinear systems: Application to economic model predictive control of nonlinear processes. *AIChE Journal*, 61:3353–3373, 2015.

[7] Anas Alanqar, Helen Durand, and Panagiotis D Christofides. Error-triggered on-line model identification for model-based feedback control. *AIChE Journal*, 63:949–966, 2017.

[8] Anas Alanqar, Matthew Ellis, and Panagiotis D Christofides. Economic model predictive control of nonlinear process systems using empirical models. *AIChE Journal*, 61:816–830, 2015.

[9] Mohammed Alhajeri, Junwei Luo, Zhe Wu, Fahad Albalawi, and Panagiotis D Christofides. Process structure-based recurrent neural network modeling for predictive control: A comparative study. *Chemical Engineering Research and Design*, 179:77–89, 2022.

[10] Mohammed Alhajeri and Masoud Soroush. Tuning guidelines for model-predictive control. *Industrial & Engineering Chemistry Research*, 59:4177–4191, 2020.

[11] Abd AlRahman R. AlMomani, Jie Sun, and Erik Bollt. How entropic regression beats the outliers problem in nonlinear system identification. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 30:013107, 2020.

[12] Shun Ichi Amari. Backpropagation and stochastic gradient descent method. *Neurocomputing*, 5:185–196, 1993.

[13] EC Amazon. Amazon web services.

[14] Rishi Amrit, James B Rawlings, and David Angeli. Economic optimization using model predictive control with a terminal cost. *Annual Reviews in Control*, 35:178–186, 2011.

[15] David Angeli, Rishi Amrit, and James B Rawlings. On average performance and stability of economic model predictive control. *IEEE Transactions on Automatic Control*, 57:1615–1626, 2011.

[16] Yiğit M Arısoy, Luis E Criales, and Tuğrul Özel. Modeling and simulation of thermal field and solidification in laser powder bed fusion of nickel alloy IN-625. *Optics & Laser Technology*, 109:278–292, 2019.

[17] Felix W Baumann and Dieter Roller. Additive manufacturing, cloud-based 3D printing and associated services—overview. *Journal of Manufacturing and Materials Processing*, 1:15, 2017.

[18] Martin Baumers, Luca Beltrametti, Angelo Gasparre, and Richard Hague. Informing additive manufacturing technology adoption: total cost and the impact of capacity utilisation. *International Journal of Production Research*, 55:6957–6970, 2017.

[19] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305, 2012.

[20] Lorenz T Biegler. *Nonlinear programming: Concepts, algorithms, and applications to chemical processes*. SIAM, 2010.

[21] Lorenz T Biegler, Yi-Dong Lang, and Weijie Lin. Multi-scale optimization for process systems engineering. *Computers & Chemical Engineering*, 60:17–30, 2014.

[22] Stephen A Billings. Identification of nonlinear systems-a survey. 127:272–285, 1980.

[23] Fabio Bonassi, Marcello Farina, Jing Xie, and Riccardo Scattolini. On Recurrent Neural Networks for learning-based control: recent results and ideas for future developments. *Journal of Process Control*, 114:92–104, 2022.

[24] Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113:3932–3937, 2016.

[25] Mark A Buckner and Lonnie J Love. Automating and accelerating the additive manufacturing design process with multi-objective constrained evolutionary optimization and hpc/cloud computing. In *2012 Future of Instrumentation International Workshop (FIIW) Proceedings*, pages 1–4, Gatlinburg, TN, 2012. IEEE.

[26] Javaid Butt. Exploring the interrelationship between additive manufacturing and industry 4.0. *Designs*, 4:13, 2020.

[27] Eduardo F Camacho and Carlos Bordons Alba. *Model Predictive Control*. Springer Science & Business Media, London, 2nd edition, 2013.

[28] Hsueh-Chia Chang and Mobolaji Aluko. Multi-scale analysis of exotic dynamics in surface catalyzed reactions I: Justification and preliminary model discriminations. *Chemical Engineering Science*, 39:37–50, 1984.

[29] Baotong Chen, Jiafu Wan, Lei Shu, Peng Li, Mithun Mukherjee, and Boxing Yin. Smart factory of industry 4.0: Key technologies, application case, and challenges. *IEEE Access*, 6:6505–6519, 2018.

[30] Hui Chen, Qingsong Wei, Yingjie Zhang, Fan Chen, Yusheng Shi, and Wentao Yan. Powder-spreading mechanisms in powder-bed-based additive manufacturing: Experiments and computational modeling. *Acta Materialia*, 179:158–171, 2019.

[31] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

[32] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. pages 103–111, Doha, Qatar, 2014. Association for Computational Linguistics.

[33] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. pages 1724–1734, Doha, Qatar, 2014.

[34] Gregory C Chow et al. *Analysis and control of dynamic economic systems*. Wiley, 1975.

[35] Tommy WS Chow and Yong Fang. A recurrent neural-network-based real-time learning control strategy applying to nonlinear systems with unknown dynamics. *IEEE Transactions on Industrial Electronics*, 45:151–161, 1998.

[36] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[37] David W Clarke and Peter J Gawthrop. Self-tuning control. *Proceedings of the Institution of Electrical Engineers*, 126:633–640, 1979.

[38] DW Clarke and PJ Gawthrop. Self-tuning controller. *Proceedings of the Institution of Electrical Engineers*, 122:929–934, 1975.

[39] Stijn Clijsters, Tom Craeghs, Sam Buls, Karolien Kempen, and J-P Kruth. In situ quality control of the selective laser melting process using a high-speed, real-time melt pool monitoring system. *The International Journal of Advanced Manufacturing Technology*, 75:1089–1101, 2014.

[40] Luis C Costa, Rui Vilar, Tamas Reti, and Augusto M Deus. Rapid tooling by laser powder deposition: Process simulation using finite element analysis. *Acta Materialia*, 53:3987–3999, 2005.

[41] Tom Craeghs, Florian Bechmann, Sebastian Berumen, and Jean-Pierre Kruth. Feedback control of layerwise laser melting using optical sensors. *Physics Procedia*, 5:505–514, 2010.

[42] Balázs Csanád Csáji et al. Approximation with artificial neural networks. *Faculty of Sciences, Etvs Lornd University, Hungary*, 24:7, 2001.

[43] Jordi Delgado, Joaquim Ciurana, and Ciro A Rodríguez. Influence of process parameters on part quality and mechanical properties for DMLS and SLM with iron-based materials. *The International Journal of Advanced Manufacturing Technology*, 60:601–610, 2012.

[44] Andrew Dickins, Taufiq Widjanarko, Simon Lawes, P Stavroulakis, and Richard Leach. Design of a multi-sensor in-situ inspection system for additive manufacturing. In *ASPE and EUSPEN Summer Topical Meeting on Advancing Precision in Additive Manufacturing*, pages 248–152, Berkeley, CA, 2018.

[45] Thomas G Dietterich. Machine learning for sequential data: A review. In *Joint IAPR international workshops on statistical techniques in pattern recognition (SPR) and structural and syntactic pattern recognition (SSPR)*, pages 15–30. Springer, 2002.

[46] Ugur M. Dilberoglu, Bahar Gharehpapagh, Ulas Yaman, and Melik Dolen. The role of additive manufacturing in the era of industry 4.0. *Procedia Manufacturing*, 11:545–554, 2017.

[47] Pedro M Domingos. Why does bagging work? A bayesian account and its implications. *KDD*,pages 155–158, 1997.

[48] Juan Peralta Donate, Paulo Cortez, German Gutierrez Sanchez, and Araceli Sanchis De Miguel. Time series forecasting using a weighted cross-validation evolutionary artificial neural network ensemble. *Neurocomputing*, 109:27–32, 2013.

[49] Zhichao Dong, Yabo Liu, Weibin Wen, Jingran Ge, and Jun Liang. Effect of hatch spacing on melt pool and as-built quality during selective laser melting of stainless steel: modeling and experimental approaches. *Materials*, 12:50, 2019.

[50] Andreas Draeger, Sebastian Engell, and Horst Ranke. Model predictive control using neural networks. *IEEE Control Systems Magazine*, 15:61–66, 1995.

[51] Sophia N Economidou and Dimitris Karalekas. Optical sensor-based measurements of thermal expansion coefficient in additive manufacturing. *Polymer Testing*, 51:117–121, 2016.

[52] Matthew Ellis, Helen Durand, and Panagiotis D. Christofides. A tutorial review of economic model predictive control methods. *Journal of Process Control*, 24:1156–1178, 2014.

[53] Matthew J Ellis and Venkatesh Chinde. An encoder–decoder LSTM-based EMPC framework applied to a building HVAC system. *Chemical Engineering Research and Design*, 160:508–520, 2020.

[54] EOS. EOSTATE MeltPool: Real-time process monitoring for EOS M 290, 2018.

[55] Erik Esche, Joris Weigert, Gerardo Brand Rihm, Jan Göbel, and Jens-Uwe Repke. Architectures for neural networks as surrogates for dynamic systems in chemical engineering. *Chemical Engineering Research and Design*, 177:184–199, 2022.

[56] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. *KDD*, 96:226–231, 1996.

[57] Sarah K Everton, Matthias Hirsch, Petros Stravroulakis, Richard K Leach, and Adam T Clare. Review of in-situ process monitoring and in-situ metrology for metal additive manufacturing. *Materials & Design*, 95:431–445, 2016.

[58] Ronan Fablet, Said Ouala, and Cédric Herzet. Bilinear residual neural network for the identification and forecasting of geophysical dynamics. In *Proceedings of the 26th European Signal Processing Conference*, pages 1477–1481, Rome, Italy, 2018.

[59] Behrouz A Forouzan. *TCP/IP protocol suite*. McGraw-Hill Higher Education, 2002.

[60] William E Frazier. Metal additive manufacturing: a review. *Journal of Materials Engineering and Performance*, 23:1917–1928, 2014.

[61] Yarin Gal and Zoubin Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. *Advances in Neural Information Processing Systems*, 29, 2016.

[62] Banning Garrett. 3D printing: New economic paradigms and strategic shifts. *Global Policy*, 5:70–75, 2014.

[63] W. Gentzsch. Sun Grid Engine: towards creating a compute power grid. In *Proceeding First IEEE/ACM International Symposium on Cluster Computing and the Grid*, pages 35–36, Brisbane, Australia, 2001.

[64] Ian Gibson, David W Rosen, Brent Stucker, et al. *Additive Manufacturing Technologies*, volume 17. Springer, 2014.

[65] Haijun Gong, Khalid Rafi, Hengfeng Gu, Thomas Starr, and Brent Stucker. Analysis of defect generation in Ti–6Al–4V parts made using powder bed fusion additive manufacturing processes. *Additive Manufacturing*, 1:87–98, 2014.

[66] Raul González-García, Ramiro Rico-Martìnez, and Ioannis G Kevrekidis. Identification of distributed parameter systems: A neural net based approach. *Computers & Chemical Engineering*, 22:S965–s968, 1998.

[67] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT press, 2016.

[68] Eberhard KU Gross and F1097 Reiner M Dreizler. *Density functional theory*, volume 337. Springer Science & Business Media, 2013.

[69] Thomas Grünberger and Robert Domröse. Optical in-process monitoring of direct metal laser sintering (DMLS): A revolutionary technology meets automated quality inspection. *Laser Technik Journal*, 11:40–42, 2014.

[70] Kevin Gurney. *An introduction to neural networks*. CRC press, 2018.

[71] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. Co-teaching: Robust training of deep neural networks with extremely noisy labels. *Advances in Neural Information Processing Systems*, 31, 2018.

[72] Quanquan Han, Heng Gu, and Rossitza Setchi. Discrete element simulation of powder layer thickness in laser additive manufacturing. *Powder technology*, 352:91–102, 2019.

[73] Ramadane Hedjar. Adaptive neural network model predictive control. *International Journal of Innovative Computing, Information and Control*, 9:1245–1257, 2013.

[74] Michael A Henson and Dale E Seborg. *Nonlinear process control*. Prentice Hall PTR Upper Saddle River, New Jersey, 1997.

[75] David M Himmelblau. Applications of artificial neural networks in chemical engineering. *Korean Journal of Chemical Engineering*, 17:373–392, 2000.

[76] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–1780, 1997.

[77] Kailas Holkar and Laxman M Waghmare. An overview of model predictive control. *International Journal of Control and Automation*, 3:47–63, 2010.

[78] John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79:2554–2558, 1982.

[79] Davor Hrovat, Stefano Di Cairano, H Eric Tseng, and Ilya V Kolmanovsky. The development of model predictive control in automotive industry: A survey. In *2012 IEEE International Conference on Control Applications*, pages 295–302, Dubrovnik, Croatia, 2012.

[80] Olga Ivanova and Thomas Campbell. Additive manufacturing as a disruptive technology: Implications of three-dimensional printing. *Technology and Innovation*, 15:67–69, 01 2013.

[81] Karolien Kempen, Evren Yasa, Lore Thijs, J-P Kruth, and Jan Van Humbeeck. Microstructure and mechanical properties of selective laser melted 18Ni-300 steel. *Physics Procedia*, 12:255–263, 2011.

[82] AM Khalil, IS Loginova, AN Solonin, and AO Mosleh. Controlling liquation behavior and solidification cracks by continuous laser melting process of AA-7075 aluminum alloy. *Materials Letters*, 277:128364, 2020.

[83] Mojtaba Khanzadeh, Sudipta Chowdhury, Mark A Tschopp, Haley R Doude, Mohammad Marufuzzaman, and Linkan Bian. In-situ monitoring of melt pool images for porosity prediction in directed energy deposition processes. *IISE Transactions*, 51:437–455, 2019.

[84] Jivtesh Khurana, Bradley Hanks, and Mary Frecker. Design for additive manufacturing of cellular compliant mechanism using thermal history feedback. In *ASME 2018 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Quebec City, Canada, 2018. American Society of Mechanical Engineers Digital Collection.

[85] Tung Kieu, Bin Yang, and Christian S Jensen. Outlier detection for multidimensional time series using deep neural networks. In *2018 19th IEEE International Conference on Mobile Data Management (MDM)*, pages 125–134, Aalborg, Denmark, 2018.

[86] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[87] Paisan Kittisupakorn, Piyanuch Thitiyasook, Mohd Azlan Hussain, and Wachira Daosud. Neural network based model predictive control for a steel pickling process. *Journal of Process Control*, 19:579–590, 2009.

[88] Charles M Kozierok. *The TCP/IP Guide: A Comprehensive, Illustrated Internet Protocols Reference*. No Starch Press, 2005.

[89] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[90] R Leal, FM Barreiros, L Alves, F Romeiro, JC Vasco, M Santos, and C Marto. Additive manufacturing tooling for the automotive industry. *The International Journal of Advanced Manufacturing Technology*, 92:1671–1676, 2017.

[91] Jay H Lee. Modeling and identification for nonlinearmodel predictive control: Requirements, current status and future research needs. In *Nonlinear model predictive control*, pages 269–293. Springer, 2000.

[92] Hirpa G Lemu. On opportunities and limitations of additive manufacturing technology for industry 4.0 era. In *International Workshop of Advanced Manufacturing and Automation*, pages 106–113, Changzhou, China, 2018. Springer.

[93] Jean Lévine and Pierre Rouchon. Quality control of binary distillation columns via nonlinear aggregated models. *Automatica*, 27:463–480, 1991.

[94] Dan Li, Dacheng Chen, Baihong Jin, Lei Shi, Jonathan Goh, and See-Kiong Ng. MAD-GAN: Multivariate anomaly detection for time series data with generative adversarial networks. In *International Conference on Artificial Neural Networks*, pages 703–716, Munich, Germany, 2019. Springer.

[95] Jian Li, Jing Hu, Yi Zhu, Xiaowen Yu, Mengfei Yu, and Huayong Yang. Surface roughness control of root analogue dental implants fabricated using selective laser melting. *Additive Manufacturing*, 34:101283, 2020.

[96] Yue Li and Zheming Tong. Model predictive control strategy using encoder-decoder recurrent neural networks for smart control of thermal environment. *Journal of Building Engineering*, 42:103017, 2021.

[97] Yuandan Lin and Eduardo D Sontag. A universal formula for stabilization with bounded controls. *Systems & Control Letters*, 16:393–397, 1991.

[98] Hancong Liu, Sirish Shah, and Wei Jiang. On-line outlier detection and data cleaning. *Computers & Chemical Engineering*, 28:1635–1647, 2004.

[99] Renwei Liu, Zhiyuan Wang, Todd Sparks, Frank Liou, and Joseph Newkirk. Aerospace applications of laser additive manufacturing. In *Laser Additive Manufacturing*, pages 351–371. Elsevier, 2017.

[100] Cody S Lough, Xin Wang, Christopher C Smith, Robert G Landers, Douglas A Bristow, James A Drallmeier, Ben Brown, and Edward C Kinzel. Correlation of SWIR imaging with LPBF 304L stainless steel part properties. *Additive Manufacturing*, 35:101359, 2020.

[101] Lu Lu, Pengzhan Jin, and George Em Karniadakis. Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *arXiv preprint arXiv:1910.03193*, 2019.

[102] Chao Ma, Madhu Vadali, Neil A Duffie, Frank E Pfefferkorn, and Xiaochun Li. Melt pool flow and surface evolution during pulsed laser micro polishing of Ti_6Al_4V. *Journal of Manufacturing Science and Engineering*, 135, 2013.

[103] Mingming Ma, Zemin Wang, Ming Gao, and Xiaoyan Zeng. Layer thickness dependence of performance in high-power selective laser melting of $Cr_{18}Ni_9Ti$ stainless steel. *Journal of Materials Processing Technology*, 215:142–150, 2015.

[104] Dougal Maclaurin, David Duvenaud, and Ryan Adams. Gradient-based hyperparameter optimization through reversible learning. In *International Conference on Machine Learning*, pages 2113–2122, Lille, France, 2015. PMLR.

[105] Yahya Mahmoodkhani, Usman Ali, Shahriar Imani Shahabad, Adhitan Rani Kasinathan, Reza Esmaeilizadeh, Ali Keshavarzkermani, Ehsan Marzbanrad, and Ehsan Toyserkani. On the measurement of effective powder layer thickness in laser powder-bed fusion additive manufacturing of metals. *Progress in Additive Manufacturing*, 4:109–116, 2019.

[106] Arfan Majeed, Yingfeng Zhang, Shan Ren, Jingxiang Lv, Tao Peng, Saad Waqar, and Enhuai Yin. A big data-driven framework for sustainable and smart additive manufacturing. *Robotics and Computer-Integrated Manufacturing*, 67:102026, 2021.

[107] Bryon R Maner and Francis J Doyle III. Polymerization reactor control using autoregressive-plus volterra-based MPC. *AIChE Journal*, 43:1763–1784, 1997.

[108] David Q Mayne, James B Rawlings, Christopher V Rao, and Pierre OM Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36:789–814, 2000.

[109] Tait D McLouth, David B Witkin, Glenn E Bean, Scott D Sitzman, Paul M Adams, Julian R Lohser, Jenn-Ming Yang, and Rafael J Zaldivar. Variations in ambient and elevated temperature mechanical behavior of IN718 manufactured by selective laser melting via process parameter control. *Materials Science and Engineering: A*, 780:139184, 2020.

[110] Larry R Medsker and LC Jain. *Recurrent neural networks: Design and Applications*. CRC press, 2001.

[111] Mehrshad Mehrpouya, Amir Dehghanghadikolaei, Behzad Fotovvati, Alireza Vosooghnia, Sattar S. Emamian, and Annamaria Gisario. The potential of additive manufacturing in the smart factory industrial 4.0: A review. *Applied Sciences*, 9:3865, 2019.

[112] Joao Mendes-Moreira, Carlos Soares, Alípio Mário Jorge, and Jorge Freire De Sousa. Ensemble approaches for regression: A survey. *ACM Computing Surveys*, 45:1–40, 2012.

[113] Milos Miljanovic. Comparative analysis of recurrent and finite impulse response neural networks in time series prediction. *Indian Journal of Computer Science and Engineering*, 3:180–191, 2012.

[114] W Thomas Miller, Richard S Sutton, and Paul J Werbos. *Neural networks for control*. MIT press, 1995.

[115] Nima Mohajerin and Steven L Waslander. Multistep prediction of dynamic systems with recurrent neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 30:3370–3383, 2019.

[116] Swati Mohanty. Artificial neural network based system identification and model predictive control of a flotation column. *Journal of Process Control*, 19:991–999, 2009.

[117] Gunther Mohr, Simon J Altenburg, Alexander Ulbricht, Philipp Heinrich, Daniel Baum, Christiane Maierhofer, and Kai Hilgenberg. In-situ defect detection in laser powder bed fusion by using thermography and optical tomography—comparison to computed tomography. *Metals*, 10:103, 2020.

[118] Manfred Morari and Jay H Lee. Model predictive control: past, present and future. *Computers & Chemical Engineering*, 23:667–682, 1999.

[119] Daniel Moser, Sreekanth Pannala, and Jayathi Murthy. Computation of effective thermal conductivity of powders for selective laser sintering simulations. *Journal of Heat Transfer*, 138:8, 2016.

[120] Peeyush Nandwana, William H Peter, Ryan R Dehoff, Larry E Lowe, Michael M Kirka, Francisco Medina, and Sudarsanam S Babu. Recyclability study on Inconel 718 and Ti-6Al-4V powders for use in electron beam melting. *Metallurgical and Materials Transactions B*, 47:754–762, 2016.

[121] Quy B Nguyen, Duy N Luu, Mui Ling Sharon Nai, Z Zhu, Z Chen, and J Wei. The role of powder layer thickness on the quality of slm printed parts. *Archives of Civil and Mechanical Engineering*, 18:948–955, 2018.

[122] Chinedum E Okwudire, Sharankumar Huggi, Sagar Supe, Chengyang Huang, and Bowen Zeng. Low-level control of 3d printers from the cloud: A step toward 3D printer control as a service. *Inventions*, 3:56, 2018.

[123] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, and Luca Antiga. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32, 2019.

[124] J Pemberton. Non-linear and non-stationary time series analysis., 1990.

[125] AD Peralta, M Enright, M Megahed, J Gong, M Roybal, and J Craig. Towards rapid qualification of powder-bed laser additively manufactured parts. *Integrating Materials and Manufacturing Innovation*, 5:154–176, 2016.

[126] Robi Polikar. Ensemble learning. In *Ensemble Machine Learning*, pages 1–34. Springer, 2012.

[127] S Joe Qin and Thomas A Badgwell. A survey of industrial model predictive control technology. *Control engineering practice*, 11:733–764, 2003.

[128] S Joe Qin and Thomas J McAvoy. Nonlinear PLS modeling using neural networks. *Computers & Chemical Engineering*, 16:379–391, 1992.

[129] Chunlei Qiu, Chinnapat Panwisawas, Mark Ward, Hector C Basoalto, Jeffery W Brooks, and Moataz M Attallah. On the role of melt flow into the surface structure and porosity development during selective laser melting. *Acta Materialia*, 96:72–79, 2015.

[130] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Multistep neural networks for data-driven discovery of nonlinear dynamical systems. *arXiv:1801.01236*, 2018.

[131] Davi Ramos, Fawzi Belblidia, and Johann Sienz. New scanning strategy to reduce warpage in additive manufacturing. *Additive Manufacturing*, 28:554–564, 2019.

[132] James B Rawlings. Tutorial overview of model predictive control. *IEEE Control Systems Magazine*, 20:38–52, 2000.

[133] Lei Ren, Lin Zhang, Fei Tao, Chun Zhao, Xudong Chai, and Xinpei Zhao. Cloud manufacturing: from concept to practice. *Enterprise Information Systems*, 9:186–209, 2015.

[134] Yi Ming Ren, Yichi Zhang, Yangyao Ding, Tao Liu, Cody S Lough, Ming C Leu, Edward C Kinzel, and Panagiotis D Christofides. Finite element modeling of direct metal laser solidification process: Sensor data replication and use in defect detection and data reduction via machine learning. *Chemical Engineering Research and Design*, 171:254–267, 2021.

[135] Yi Ming Ren, Yichi Zhang, Yangyao Ding, Yongjian Wang, and Panagiotis D Christofides. Computational fluid dynamics-based in-situ sensor analytics of direct metal laser solidification process using machine learning. *Computers & Chemical Engineering*, 143:107069, 2020.

[136] Matheus Henrique Dal Molin Ribeiro, Stéfano Frizzo Stefenon, José Donizetti de Lima, Ademir Nied, Viviana Cocco Mariani, and Leandro dos Santos Coelho. Electricity price forecasting based on self-adaptive decomposition and heterogeneous ensemble learning. *Energies*, 13:5190, 2020.

[137] John Romano, Leila Ladani, and Magda Sadowski. Laser additive melting and solidification of Inconel 718: finite element simulation and experiment. *JOM*, 68:967–977, 2016.

[138] Daniel Rosenthal. Mathematical theory of heat distribution during welding and cutting. *Welding Journal*, 20:220–234, 1941.

[139] Yulia Rubanova, Ricky TQ Chen, and David K Duvenaud. Latent ordinary differential equations for irregularly-sampled time series. *Advances in neural information processing systems*, 32, 2019.

[140] Samuel H. Rudy, J. Nathan Kutz, and Steven L. Brunton. Deep learning of dynamics and signal-noise decomposition with time-stepping constraints. *Journal of Computational Physics*, 396:483–506, 2019.

[141] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[142] Magda Sadowski, Leila Ladani, William Brindley, and John Romano. Optimizing quality of additively manufactured Inconel 718 using powder bed laser melting process. *Additive Manufacturing*, 11:60–70, 2016.

[143] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[144] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45:2673–2681, 1997.

[145] Luke Scime and Jack Beuth. Using machine learning to identify in-situ melt pool signatures indicative of flaw formation in a laser powder bed fusion additive manufacturing process. *Additive Manufacturing*, 25:151–165, 2019.

[146] Luke Scime, Derek Siddel, Seth Baird, and Vincent Paquit. Layer-wise anomaly detection and classification for powder bed additive manufacturing processes: A machine-agnostic algorithm for real-time pixel-wise semantic segmentation. *Additive Manufacturing*, 36:101453, 2020.

[147] Pierre OM Scokaert, David Q Mayne, and James B Rawlings. Suboptimal model predictive control (feasibility implies stability). *IEEE Transactions on Automatic Control*, 44:648–654, 1999.

[148] Clare Scott. EOS introduces EOSTATE Exposure OT, First Commercial Optical Tomography System for Additive Manufacturing, 2017.

[149] M Hossein Sehhat and Ali Mahdianikhotbesara. Powder spreading in laser-powder bed fusion process. *Granular Matter*, 23:1–18, 2021.

[150] Sunpreet Singh and Seeram Ramakrishna. Biomedical applications of additive manufacturing: present and future. *Current Opinion in Biomedical Engineering*, 2:105–115, 2017.

[151] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *Advances in Neural Information Processing Systems*, 25, 2012.

[152] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15:1929–1958, 2014.

[153] W Richard Stevens and Gary R Wright. *TCP/IP illustrated, volume 2*. Addison-Wesley, 1996.

[154] V Sh Sufiiarov, AA Popovich, EV Borisov, IA Polozov, DV Masaylo, and AV Orlov. The effect of layer thickness at selective laser melting. *Procedia engineering*, 174:126–134, 2017.

[155] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in Neural Information Processing Systems*, 27, 2014.

[156] Paulo Tabuada. Event-triggered real-time scheduling of stabilizing control tasks. *IEEE Transactions on Automatic Control*, 52:1680–1685, 2007.

[157] Ming Tang, P Chris Pistorius, and Jack L Beuth. Prediction of lack-of-fusion porosity for powder bed fusion. *Additive Manufacturing*, 14:39–48, 2017.

[158] Saurabh Vaidya, Prashant Ambad, and Santosh Bhosle. Industry 4.0 – a glimpse. *Procedia Manufacturing*, 20:233–238, 2018.

[159] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[160] Xiaofeng Wang and Michael D Lemmon. Event design in event-triggered feedback control systems. In *2008 47th IEEE Conference on Decision and Control*, pages 2105–2110, Cancun, Mexico, 2008.

[161] Yuanbin Wang, Yuan Lin, Ray Y Zhong, and Xun Xu. IoT-enabled cloud-based additive manufacturing platform to support rapid product development. *International Journal of Production Research*, 57:3975–3991, 2019.

[162] Jordan S Weaver, Justin Whiting, Vipin Tondare, Carlos Beauchamp, Max Peltz, Jared Tarr, Thien Q Phan, and M Alkan Donmez. The effects of particle size distribution on the rheological properties of the powder and the mechanical properties of additively manufactured 17-4 PH stainless steel. *Additive Manufacturing*, 39:101851, 2021.

[163] P Eo Wellstead, D Prager, and P Zanker. Pole assignment self-tuning regulator. In *Proceedings of the Institution of Electrical Engineers*, volume 126, pages 781–787. IET, 1979.

[164] Paul J Werbos. Backpropagation through time: What it does and how to do it. *Proceedings of the IEEE*, 78:1550–1560, 1990.

[165] Zachary T Wilson and Nikolaos V Sahinidis. The ALAMO approach to machine learning. *Computers & Chemical Engineering*, 106:785–795, 2017.

[166] Tim Marten Wischeropp, Claus Emmelmann, Milan Brandt, and Aaron Pateras. Measurement of actual powder layer height and packing density in a single layer in selective laser melting. *Additive Manufacturing*, 28:176–183, 2019.

[167] Kaufui V Wong and Aldo Hernandez. A review of additive manufacturing. *International scholarly research notices*, 2012:208760, 2012.

[168] Wee Chin Wong, Ewan Chee, Jiali Li, and Xiaonan Wang. Recurrent neural network-based model predictive control for continuous pharmaceutical manufacturing. *Mathematics*, 6:242, 2018.

[169] Zhe Wu, Aisha Alnajdi, Quanquan Gu, and Panagiotis D Christofides. Statistical machine-learning-based predictive control of uncertain nonlinear processes. *AIChE Journal*, 68:e17642, 2022.

[170] Zhe Wu and Panagiotis D Christofides. *Process operational safety and cybersecurity: A feedback control approach.* Springer, 2021.

[171] Zhe Wu, Junwei Luo, David Rincon, and Panagiotis D Christofides. Machine learning-based predictive control using noisy data: Evaluating performance and robustness via a large-scale process simulator. *Chemical Engineering Research and Design*, 168:275–287, 2021.

[172] Zhe Wu, David Rincon, and Panagiotis D Christofides. Real-time adaptive machine-learning-based predictive control of nonlinear processes. *Industrial & Engineering Chemistry Research*, 59:2275–2290, 2019.

[173] Zhe Wu, David Rincon, and Panagiotis D Christofides. Process structure-based recurrent neural network modeling for model predictive control of nonlinear processes. *Journal of Process Control*, 89:74–84, 2020.

[174] Zhe Wu, David Rincon, Quanquan Gu, and Panagiotis D Christofides. Statistical machine learning in model predictive control of nonlinear processes. *Mathematics*, 9:1912, 2021.

[175] Zhe Wu, Anh Tran, Yi Ming Ren, Cory S Barnes, Scarlett Chen, and Panagiotis D Christofides. Model predictive control of phthalic anhydride synthesis in a fixed-bed catalytic reactor via machine learning modeling. *Chemical Engineering Research and Design*, 145:173–183, 2019.

[176] Zhe Wu, Anh Tran, David Rincon, and Panagiotis D Christofides. Machine learning-based predictive control of nonlinear processes. part I: theory. *AIChE Journal*, 65:e16729, 2019.

[177] Fangda Xu, Vimal Dhokia, Paul Colegrove, Anthony McAndrew, Stewart Williams, Andrew Henstridge, and Stephen T Newman. Realisation of a multi-sensor framework for process monitoring of the wire arc additive manufacturing in producing Ti-6Al-4V parts. *International Journal of Computer Integrated Manufacturing*, 31:785–798, 2018.

[178] Jing Xu, Chuandong Li, Xing He, and Tingwen Huang. Recurrent neural network for solving model predictive control problem in application of four-tank benchmark. *Neurocomputing*, 190:172–178, 2016.

[179] Min Xu, Jeanne M David, Suk Hi Kim, et al. The fourth industrial revolution: Opportunities and challenges. *International journal of financial research*, 9:90–95, 2018.

[180] Yuhui Xu, Lingxi Xie, Wenrui Dai, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Hongkai Xiong, and Qi Tian. Partially-connected neural architecture search for reduced computational redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43:2953–2970, 2021.

[181] Wentao Yan, Yanping Lian, Cheng Yu, Orion L Kafka, Zeliang Liu, Wing Kam Liu, and Gregory J Wagner. An integrated process–structure–property modeling framework for additive manufacturing. *Computer Methods in Applied Mechanics and Engineering*, 339:184–204, 2018.

[182] Jingjing Yang, Hanchen Yu, Jie Yin, Ming Gao, Zemin Wang, and Xiaoyan Zeng. Formation and control of martensite in Ti-6Al-4V alloy produced by selective laser melting. *Materials & Design*, 108:308–318, 2016.

[183] Shen Yin and Okyay Kaynak. Big data for modern industry: Challenges and trends, point of view. *Proceedings of the IEEE*, 103:143–146, 2015.

[184] Yohan Yoon, Jungryoul Yim, Eunho Choi, Junghan Kim, Kyuhwan Oh, and Youngchang Joo. Texture control of 3.04%-si electrical steel sheets by local laser melting and directional solidification. *Materials Letters*, 185:43–46, 2016.

[185] Bodi Yuan, Brian Giera, Gabe Guss, Ibo Matthews, and Sara Mcmains. Semi-supervised convolutional neural networks for in-situ video monitoring of selective laser melting. In *Proceedings of Winter Conference on Applications of Computer Vision (WACV)*, pages 744–753, Waikoloa Village, Hawaii, 2019. IEEE.

[186] Krzysztof Zarzycki and Maciej Ławryńczuk. LSTM and GRU neural networks as models of dynamical processes used in predictive control: A comparison of models developed for two chemical reactors. *Sensors*, 21:5625, 2021.

[187] Hossein Ghaderi Zefrehi and Hakan Altınçay. Imbalance learning using heterogeneous ensembles. *Expert Systems with Applications*, 142:113005, 2020.

[188] Aston Zhang, Zachary C Lipton, Mu Li, and Alexander J Smola. Dive into deep learning. *arXiv preprint arXiv:2106.11342*, 2021.

[189] Bo Zhang, Guojian Zou, Dongming Qin, Yunjie Lu, Yupeng Jin, and Hui Wang. A novel encoder-decoder model based on read-first LSTM for air pollutant prediction. *Science of The Total Environment*, 765:144507, 2021.

[190] Yingjie Zhang, Geok Soon Hong, Dongsen Ye, Kunpeng Zhu, and Jerry YH Fuh. Extraction and evaluation of melt pool, plume and spatter information for powder-bed fusion AM process monitoring. *Materials & Design*, 156:458–469, 2018.

[191] Zhidong Zhang, Usman Ali, Yahya Mahmoodkhani, Yuze Huang, Shahriar Imani Shahabad, Adhitan Rani Kasinathan, and Ehsan Toyserkani. Experimental and numerical investigation on the effect of layer thickness during laser powder-bed fusion of stainless steel 17-4PH. *International Journal of Rapid Manufacturing*, 9:212–230, 2020.

[192] Zhihao Zhang, Zhe Wu, David Rincon, and Panagiotis D Christofides. Real-time optimization and control of nonlinear processes using machine learning. *Mathematics*, 7:890, 2019.

[193] Yingzhe Zheng, Xiaonan Wang, and Zhe Wu. Machine learning modeling and predictive control of the batch crystallization process. *Industrial & Engineering Chemistry Research*, 61:5578–5592, 2022.

[194] Hubert Zimmermann. OSI reference model - the ISO model of architecture for open systems interconnection. *IEEE Transactions on communications*, 28:425–432, 1980.

[195] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67:301–320, 2005.