12-8-2023

An investigation into applications of canonical polyadic decomposition & ensemble learning in forecasting thermal data streams in direct laser deposition processes

Jonathan Storey
Mississippi State University, jks263@msstate.edu

An investigation into applications of canonical polyadic decomposition & ensemble learning in forecasting thermal data streams in direct laser deposition processes

By

Jonathan Storey

Approved by:

Linkan Bian (Major Professor)
Hyeona Lim
J. Edward Swan II
Matthew Priddy
Adrian Sescu (Graduate Coordinator)
Jason M. Keith (Dean, James Worth Bagley College of Engineering)

A Dissertation
Submitted to the Faculty of
Mississippi State University
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy
in Computational Engineering
in the Computational Engineering Program

Mississippi State, Mississippi

December 2023

Name: Jonathan Storey

Date of Degree: December 8, 2023

Institution: Mississippi State University

Major Field: Computational Engineering

Major Professor: Linkan Bian

Title of Study: An investigation into applications of canonical polyadic decomposition & ensemble learning in forecasting thermal data streams in direct laser deposition processes

Pages of Study: 108

Candidate for Degree of Doctor of Philosophy

Additive manufacturing (AM) is a process of creating objects from 3D model data by adding layers of material. AM technologies present several advantages compared to traditional manufacturing technologies, such as producing less material waste and being capable of producing parts with greater geometric complexity. However, deficiencies in the printing process due to high process uncertainty can affect the microstructural properties of a fabricated part leading to defects. In metal AM, previous studies have linked defects in parts with melt pool temperature fluctuations, with the size of the melt pool and the scan pattern being key factors associated with part defects. Thus being able to adjust certain process parameters during a part's fabrication, and knowing when to adjust these parameters, is critical to producing reliable parts. To know when to effectively adjust these parameters it is necessary to have models that can both identify when a defect has occurred and forecast the behavior of the process to identify if a defect will occur. This study focuses on the development of accurate forecasting models of the melt pool temperature distribution. Researchers

at Mississippi State University have collected in-situ pyrometer data of a direct laser deposition process which captures the temperature distribution of the melt pool. The high-dimensionality and noise of the data pose unique challenges in developing accurate forecasting models. To overcome these challenges, a tensor decomposition modeling framework is developed that can actively learn and adapt to new data. The framework is evaluated on two datasets which demonstrates its ability to generate accurate forecasts and adjust to new data.

Key words: additive manufacturing, thermal history forecasting, tensor decomposition, ensemble learning

DEDICATION

To my wife and son, who have brought immeasurable joy to my life. To my parents, who instilled in me a competitive spirit and love of learning. And to my Lord and Savior, for "I once was lost, but now I'm found." Soli Deo gloria.

ACKNOWLEDGEMENTS

I am indebted to many people, without whom completion of this work would not have been possible. To my wife, who is a much better writer than I, I am thankful for her proofreading. I am grateful to Dr. Linkan Bian for his guidance throughout this process. For Dr. Hyeona Lim and Dr. Ed Swan, I am thankful for the knowledge they imparted either through courses taken or in opportunities I had to work with them on various projects. A special thanks to Dr. Matthew Priddy for agreeing to serve on my committee after the tragic passing of Dr. Michael Hamilton. I had the opportunity to work with Michael for several years. He was one of the most encouraging people I've ever met, and I'm thankful for the time I had to work with him. I am grateful to the whole ISER team, especially Dr. Reed Mosher, Patti Duett, TC Falls, and Randy Jones, for their support and encouragement in my continuing education. This research was conducted by Mississippi State University under contract to the U.S. Department of Defense (DoD) High Performance Computing Modernization Program, through the US Army Engineering Research and Develop Center (ERDC) contract W912HZ21C0011. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the U.S. Army ERDC or the U.S. DoD.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Additive manufacturing (AM) is a process of creating objects from 3D model data by adding layers of material. This is in contrast to traditional manufacturing methods, such as formative manufacturing and subtractive manufacturing, which use dies or molds to inject a liquid material then allowing it to solidify, or removing material from a block to produce the desired geometry [5]. Popularly called 3D printing, and what used to be designated as rapid prototyping, AM is often discussed as a disruptive technology, changing the way products are designed and businesses developed [28].

AM technologies present several advantages compared to traditional manufacturing technologies. AM is a more efficient and less labor-intensive process than traditional manufacturing, as it does not require the same level of manual labor or complex supply chain management [9]. AM processes can reduce raw material waste by nearly 40% compared to traditional manufacturing with the potential to recycle up to 98% of the waste material [66]. Furthermore, AM can create objects with complex geometries that would be difficult or impossible to create using traditional methods, can reduce lead times by producing parts directly from 3D models without the need for tooling or machining, and can reduce costs by eliminating the need for tooling and by reducing material waste [86].

This has led AM to become increasingly popular over the past few decades being used in a wide array of industries. In the construction industry, cementitious material extrusion has been used to construct buildings, for instance a $2,600$ ft^2 single-story office building, that reduced labor by as much as 80% and construction waste by as much as 60% compared to traditional construction methods [12]. In the biomedical industry, a variety of AM technologies have been used to produce implants, including artificial valves, stents, crowns, and spinal implants [78]. In the aerospace industry, Boeing has installed thousands of AM parts on their military and commercial aircraft, while GE has integrated AM parts into critical systems like the CFM LEAP turbofan engine [29]. The use of AM technologies in the aerospace industry is expected to create $58 billion to $116 billion in economic value by 2025 [3]. While AM technologies have already had a significant impact, and are only expected to have greater impact, several barriers exist that prevent wider adoption of these technologies.

1.1 Problem Statement

More widespread adoption of AM technologies, particularly of metal-based AM, is hindered by challenges such as process uncertainty. This can be caused by a number of factors, including the properties of the material being used, the geometry of the part, and the parameters of the AM machine. This variability can lead to microstructural defects in the manufactured part, namely porosity and lack of fusion. Such defects lead to decreased reliability of the fabricated part [86, 75]. Studies have shown that the thermal history during an AM process is a strong predictor of the microstructural properties of the part, with the thermal history being affected by both the thermal history of the previous layers as well as process parameters of the AM machine [88, 64].

The mechanical properties of layers within an AM part are also correlated with the thermal history of previous layers [27]. Therefore, accurate models of the thermal behavior in an AM process are essential for understanding the mechanical properties of the fabricated parts and enabling closed loop control within an AM process.

Research into thermal models of AM processes can be broadly classified into two types: simulation-based [68, 62, 50, 25, 92, 21, 16, 26] and data-driven [67, 2, 60, 63, 35, 93, 48]. Simulation-based models can offer accurate estimates of the thermal behavior over the entire AM process, enabling identification of root causes of microstructural variations, but at significant computational cost with simulations requiring days to weeks to run [43, 54]. Given the current constraints, in some instances it may be faster to additively manufacture and experimentally test the part characteristics rather than produce simulations, making them infeasible for in-situ monitoring [73].

Data-driven models, while generally offering less accurate and more temporally limited estimates than simulation-based models, are a computationally cheaper approach that enables them to be used for in-situ monitoring. Recent studies employing data-driven approaches have used anomaly detection processes and thermal history forecasting [37, 20, 6, 45, 34, 15, 59]. Research using anomaly detection methods has focused on extracting features from thermal data streams to train supervised classification models with training labels obtained from post-process X-ray CT scanning. While these approaches have been successful, they have not demonstrated extensibility of the supervised models. Moreover, the researchers did not show that the anomaly predictors were consistent throughout the printing process. As AM processes are dynamic, and the thermal

behavior of the process can change over time, it is essential to account for potential temporal effects in the testing procedures.

In thermal history forecasting, researchers have focused on extracting spatiotemporal patterns from thermal data streams of previous layers to predict the thermal behavior of future layers. In contrast to anomaly detection, which only uses features of thermal data streams to identify defects, thermal history forecasting can be used to predict microstructure and mechanical properties of future layers, as well as to identify anomalies using control chart systems. Recent studies have demonstrated models that can forecast the process with high accuracy. However, the range of these forecasts is limited to one layer or less. Additionally, the granularity of the data is reduced in preprocessing via summaries of the Heat-Affected Zones (HAZs) or significantly smaller subsets of the HAZs. Even with the reduction in the amount of data, the models were still too computationally expensive to be used practically in an online manner.

1.2 Research Objectives, Assumptions, and Limitations

In this research, a data-driven approach to forecasting thermal behavior is introduced that uses tensor rank decomposition and ensemble learning. As shown in Khanzadeh et al. [45], a tensor formulation of the thermal history is effective in exploiting multiway dependence of layers as well as reducing dimensionality of the data. In particular, Canonical Polyadic Decomposition (CPD) is used as the formulation yields itself to capturing spatiotemporal correlations using a separable, multiplicative framework which offers a computationally efficient and highly scalable process. Methodology is developed that enables the model to be effectively online, and to transfer

a learned decomposition to data from a new process. The technical contributions of this study are summarized as follows:

- This research proposes a novel, data-driven approach to forecast melt pool temperature distribution. A 4D and 3D tensor representation of the thermal history is used to represent spatial behavior, single layer behavior, and multilayer behavior. A rank decomposition is applied to the tensor, reducing dimensionality of the data and noise within the data, enabling the construction of low-cost, low-dimensional models. The ensemble of these models enables accurate, multilayer forecasts.
- An update procedure for the 3D tensor representation is developed that enables fast adjustment to learned decomposition parameters as new data becomes available. This is accomplished in two steps: an intra-layer update which fixes the spatial characteristics and solely adjusts the temporal modes to fit incoming data, and an inter-layer update that employees alternating least squares with initialization of parameters set to previously learned modes.
- Building on the update procedure is a transfer learning procedure that enables a learned decomposition to be adjusted to data from a new process. A decomposition learned from a previous process is iteratively replaced by incoming data where the adjustments to new data propagate through the learned decomposition.
- The proposed methodologies are validated using two experimental datasets of thin wall builds from a DLD process. The experiments conducted to evaluate the methodologies indicate both high accuracy and low computational cost, making this process feasible for in-situ monitoring.

There are several limitations to this research. The methodologies proposed are experimentally validated on data collected from an individual machine (OPTOMEC LENS^{TM} 750), using a particular powder (Ti-6Al-4V powder), with fabricated parts having the same, simplistic geometry (thin wall). Certain aspects of the research, such as selection of optimal rank and memory of the tensor model, may vary if the melt pool possesses more *forms*, that is, there is more geometric variation in the melt pool. Additionally, a part with greater geometric complexity would require preprocessing of the thermal images, specifically, rotating images to align the traversal direction. However, the introduced methodology is quite robust providing much flexibility in

model selection for the ensemble process, and provided the data itself is low-rank, capable of accurately approximating the data.

1.3 Dissertation Outline

The rest of this paper is organized as follows. Chapter II provides an overview of AM technologies, distinctive properties of direct laser deposition (DLD), and survey of the literature on data-driven models of DLD processes using thermal data. Chapter III describes the methodology used and developed in this research. Chapter IV reports the results of the experiments conducted to evaluate the methodology. Lastly, Chapter V summarizes the findings in this research and outlines opportunities for future work.

CHAPTER II

LITERATURE REVIEW

2.1 An Overview of AM Technologies

The ASTM has identified seven categories of AM technologies: binder jetting, directed energy deposition (DED), material extrusion, material jetting, powder bed fusion, sheet lamination, and vat photopolymerization [5]. Figure 2.1 provides a hierarchy of the categories of AM technologies with particular instances of the listed categories.

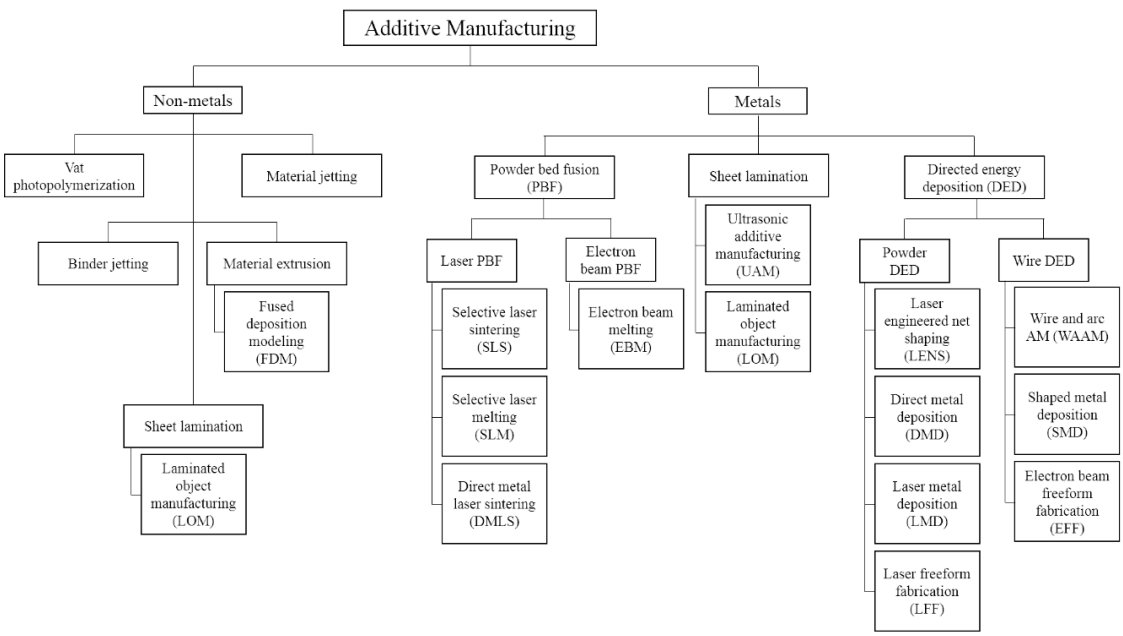

Figure 2.1

A hierarchy of AM technologies given in Yusuf et al. [57].

The different technologies vary in the types of materials they utilize and the particular benefits they provide. Binder jetting uses a liquid bonding agent to join powder particles such as from ceramics, metals, or composites. Binder jetting can be faster and more cost effective than many other AM technologies with a large amount of flexibility on the materials it can use, however, it suffers from poor mechanical properties caused by significant porosity [95]. Directed energy deposition (DED) fuses metals, either powder or wire, by melting via a heat source such as a laser or electron beam as the material is deposited. DED has many advantages including producing arbitrary shapes on even and uneven substrates, fabrication of heterogeneous materials via selectively supplying different powder or wires, and using hybrid configurations of DED with different manufacturing processes. Nevertheless, part reliability due to process uncertainty remains a significant challenge [1]. Material extrusion deposits a heated filament of a composite or thermoplastic through a nozzle, or a material that uses a chemical agent to cause solidification. Material extrusion is one of the most popular AM technologies with Fused Deposition Modeling (FDM), an extrusion-based technology, in particular being ubiquitous. FDM offers low costs, but can suffer from slower build speeds, accuracy, and material density [30]. Material jetting, much like a traditional ink printer, deposits droplets of the build material, such as a photopolymer or wax, onto the build surface which are then either hardened with UV light or cured. Advantages of material jetting include high dimensional accuracy and low surface roughness compared to other polymeric material AM technologies, while disadvantages include high sensitivity to various parameters, such as tray location and layer thickness, on mechanical properties and dimensional accuracy [33].

Powder bed fusion (PBF), similar to DED, uses a heat source such as a laser or electron beam to melt a metal or polymer powder. Different from DED though, a powder bed is spread evenly

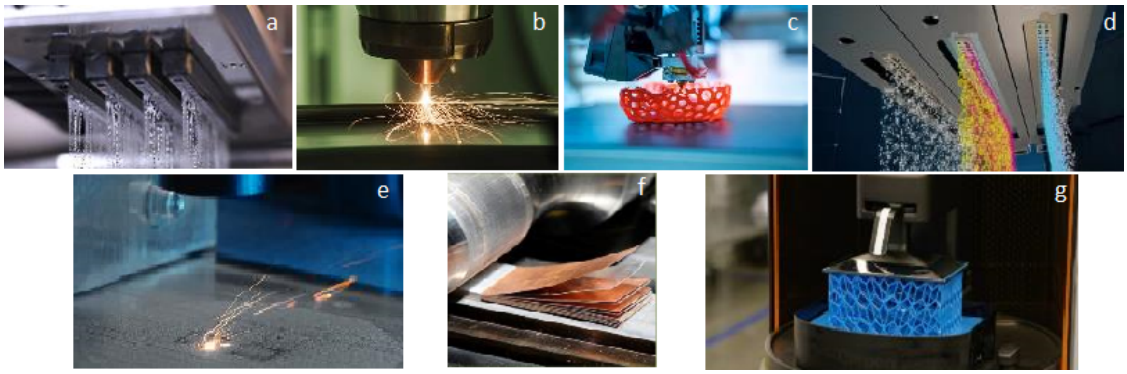

Figure 2.2

Examples of the seven categories of AM technologies [22, 84, 4, 79, 42, 83, 71].

across a build platform where the heat source selectively melts regions of the powder bed. This powder bed allows for constructing polymer parts without support structures, though supports are necessary for most metal PBF processes [31]. Compared to DED, PBF is better at building smaller, more complex parts which have a better surface finish. However, PBF is a slower process, and DED generally produces parts with better mechanical properties [85]. Like DED, PBF also suffers from reliability concerns due to process uncertainty [18]. Sheet lamination uses sheets of precisely cut material such as ceramics, metals, or paper, bonding the sheets via an adhesive or ultrasonic welding. Sheet lamination can fabricate larger parts with faster production rates than most other AM techniques, but the geometric dimensions of fabricated parts for metals can be hard to control as layers accumulate since layer thickness adjusts during consolidation under pressure [94]. Vat photopolymerization uses an electron beam or UV light to selectively cure a liquid photopolymer. Two key benefits of vat photopolymerization are the high accuracy and finish of the fabricated parts. But the photopolymers have poor impact strength and durability, and the mechanical properties degrade with age [32].

Figure 2.2 provides depictions of the seven categories of AM technologies: (a) binder jetting, (b) DED, (c) material extrusion, (d) material jetting, (e) PBF, (f) sheet lamination, and (g) vat photopolymerization. The reader can refer to Wong and Hernandez [89], Yusuf et al. [57], Ngo et al. [61], and Zhang et al. [94] for overviews about the particular technologies. Additionally, the reader can refer to Gibson et al. [30] for an exhaustive overview. Many recent studies have focused on process control methods, particularly in metal-based AM [17, 11, 38, 49, 14, 56, 8]. The next sections provide an overview of research specific to the particular AM technology used in this study.

2.1.1 Direct Laser Deposition

Of particular concern in this study is Direct Laser Deposition (DLD) as the experimental data used to test the proposed methodology comes from a DLD process. DLD is a subclass of laser-based additive manufacturing (LBAM) technologies and of DED processes. While there are both blown-powder and wire feedstock DLD processes, blown powder is a more common technology. Laser Engineered Net Shaping (LENS) is a proprietary DLD blown-powder technology that utilizes in-situ laser melting to fuse a powder material by blowing it into the beam via a nozzle(s). Due to its commercial success, LENS is the most common blown-powder DLD process within research and industry [81]. Figure 2.3 provides an illustration of the DLD process with thermal monitoring.

There are many advantages to using DLD such as high deposition rates and the ability to fabricate larger items compared to other metal-based AM methods. Additionally, DLD is a suitable technology for repairing valuable components because it creates a relatively small Heat Affected Zone (HAZ), has minimal effect on the component such as distortion and micro-cracking,

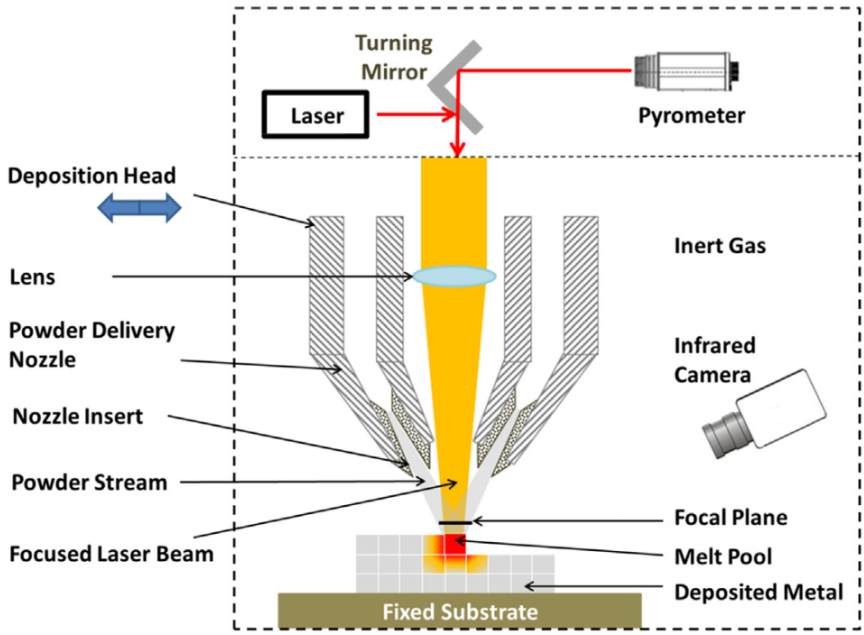

Figure 2.3

Blown powder DLD with thermal monitoring [81].

has excellent density and metallurgical bonding, and allows for precise deposition. However, as noted earlier with DED technologies in general, process uncertainty remains an issue. Pursuit of closed-loop control in DLD is of great importance as it can help reduce process uncertainty and lead to greater reliability with fabricated parts. The melt pool morphology is an aspect of a DLD process that has ramifications on the material properties of a fabricated part. As noted in Shamsaei et al. [75],

> The melt pool morphology, while in its liquid phase, is paramount to the integrity and shape of each solidified track/layer... Due to bulk heating effects and other variables, the melt pool can elongate, shrink, splash and/or become excessively superheated and unstable. To ensure consistent melt pool morphology during DLD and for each deposition layer, process parameters should be varied appropriately. The challenge is detecting variation in melt pool size/temperature and ensuring that the DLD machine automatically and effectively responds to such changes.

Shamsaei et al. describe the relationships among process parameters, thermal history, microstructure, and fatigue behavior during the DLD process as given in Figure 2.4.

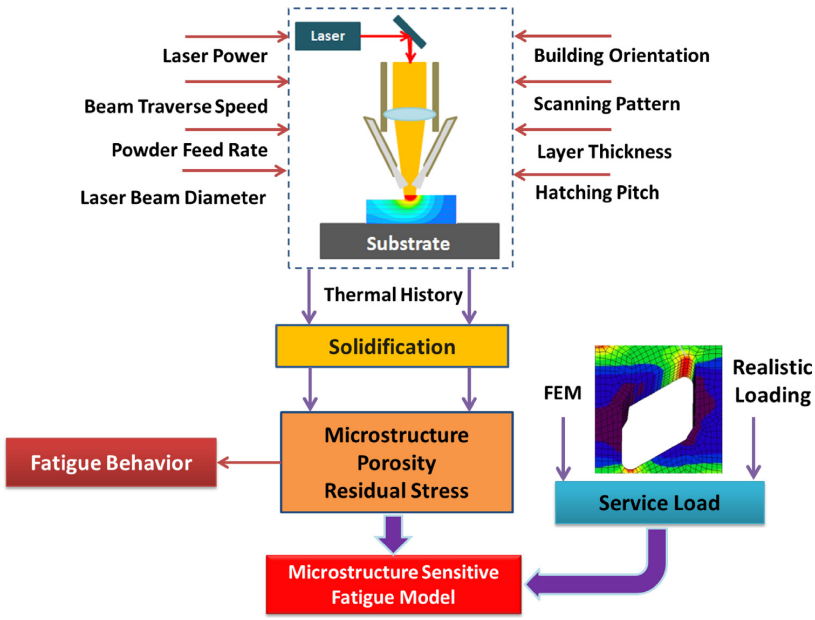

Figure 2.4

Relationships among process parameters, thermal history, microstructure, and fatigue behavior during the DLD process [75].

In their conclusion, Shamsaei et al. observe,

> ...process parameters affect the cooling rate, thermal gradients, and generally, the thermal history of DLD parts. The complex thermal history of the deposited part governs solidification, and consequently, the resultant microstructure, porosity, and residual stress formation. Mechanical properties, especially fatigue resistance, are extremely sensitive to microstructure, porosity, and residual stress within DLD components.Therefore, microstructure sensitive fatigue models, relating the microstructural features to fatigue resistance of material, can be readily employed to predict the fatigue life of DLD parts. Such analytical models can be complemented by using finite element analysis (FEA), in which critical elements with higher stresses/strain and possibility of fatigue failure can be determined.

The thermal history of a fabricated part, and in particular the thermal history of the melt pool, is of critical importance in understanding the quality of the part. Thus, *fast and accurate models of the thermal history are necessary in achieving successful closed-loop systems which will lead to greater reliability in DLD processes*. Yavari et al. [72] effectively captured this significance in their illustration given in Figure 2.5.

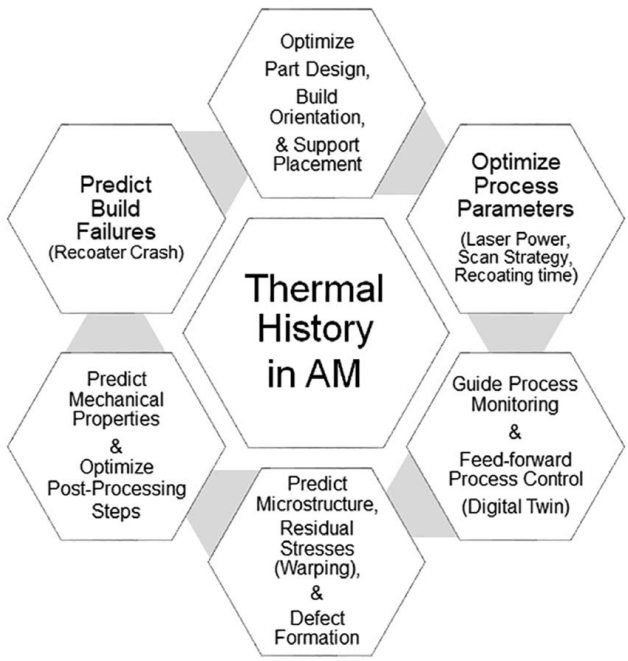

Figure 2.5

Significance of fast and accurate thermal history models for quality assurance [72].

2.2 Data-Driven Models of DLD Processes Using Thermal Data

Much research has been focused on extracting features from the melt pool thermal data to train supervised classification models to identify layer or image specific anomalies where labeling information is obtained using post-process techniques, namely X-ray computed tomography [44,

74, 37, 20, 6]. While many of the developed models can accurately identify defects, a challenge remains as to whether the models can be extended to different parts with varied geometries and materials. Additionally, research has been done that predicts mechanical properties of the part as a function of the thermal history of the part [90, 23]. Lastly, some research has sought to model the melt pool thermal history directly by capturing spatiotemporal correlations that are used to forecast the thermal history in the AM process [34, 45]. These models have been limited in the range of forecast, a single time step or one layer, and have significantly reduced the granularity of the data before modeling to ease computational burden.

2.2.1 Anomaly Detection Models using Melt Pool Thermal Images

Khanzadeh et al. [44] developed a real-time porosity prediction method based on the morphological characteristics of the melt pool boundary during a DLD process. The researchers extracted the boundary of each melt pool then employed functional principal component analysis (FPCA) for feature extraction. The extracted features were then used to train four different classification models: K-Nearest Neighbors (KNN), Support Vector Machines (SVM), Decision Trees (DT). and Linear Discriminant Analysis (LDA). The researchers found that KNNs produced the highest recall rate[1] at 98.44% while DT gave the lowest false positive rate[2] of 0.033%. A limitation of this approach, however, was that the FPCA algorithm required access to all images in the dataset to find optimal principal components for the classification methods.

Using the same data as Khanzadeh, Seifi et al. [74] developed a layer-wise process signature model to identify defect distribution per layer. The researchers cropped each 752×480 image

[1] $\frac{TP}{FN+TP}$ where TP is true positive and FN is false negative.
[2] $\frac{FP}{TN+FP}$ where FP is false positive and TN is true negative.

to 130×130 to capture the melt pool and HAZs. Each cropped image was then converted from Cartesian coordinates to spherical coordinates and biharmonic interpolation was used, reducing the final data size to 27×32. After this data transformation, Multilinear Principal Components Analysis (MPCA) was used to extract features per layer. The convex hull of these features were calculated for each layer, and the volume of the convex hulls and the maximum norm of the residuals per image for each layer from the principal component reconstruction were used as predictors of layer defects, where a layer a was identified as defected if it contained at least one pore. An SVM model was trained and tested, producing a 92% ($\frac{24}{26}$) recall rate and false positive rate of 3% ($\frac{1}{34}$).

Ho et al. [37] used Deep Learning based porosity prediction for Additive Manufacturing (DLAM) methods for real-time porosity prediction using the same data as Khanzadeh and Seifi. The researchers used 200×200 cropped images containing the melt pool and HAZs and trained several DLAMs to identify porosity per image. Several models were trained from scratch using convolutional neural networks (CNN), recurrent CNNs (RCNN), and residual RCNNs (Res-RCNN) while some pretrained models were adjusted to the new data using transfer learning. The researchers found that the Res-RCNN had the best performance in nearly all metrics with a recall of 100% ($\frac{18}{18}$) and a false positive rate of 0.5% ($\frac{2}{372}$).

Esfahani et al. [20] developed a layer-wise anomaly detection method using in-situ thermal data of a DLD process. The researchers combined an image registration method to characterize layer-wise dynamics and a Gaussian Process (GP) model to characterize variation left unexplained by the image registration operation. Features extracted from the image registration procedure and the GP model were used to train a support vector machine (SVM) classifier. The researchers tested the methodology on a single-track thin wall construction and cylinder construction using

Ti-6AL-4V powder. Layer-wise computation of the features and classification were both fast and accurate, requiring less time than that needed to build a layer and achieving an accuracy of 93.92% for the thin wall and 84.29% for the cylinder.

Bappy et al. [6] also developed methodology for layer-wise anomaly detection in which they compared results to Esfahani [20]. The researchers applied a segmentation procedure to thermal images, extracting the melt pool and the HAZs, to analyze layer-by-layer morphological dynamics. Two developed metrics, a global transition metric and a morphological transition metric, were used to characterize a layer process. Values from these metrics were used as features to train an SVM classifier. As in Esfahani [20], a single-track thin wall construction and cylinder construction were used to evaluate the methodology. Layer-wise computation of the features and classification were both fast and accurate, requiring less time than that needed to build a layer and achieving an accuracy of 96.38% for the thin wall and 85.07% for the cylinder.

2.2.2 Mechanical Property Models Using Thermal History

Researchers have demonstrated the feasibility of using the thermal history of the AM process as a means of predicting the mechanical properties of the manufactured part. Xie et al. [90] developed a mechanical properties model based on wavelet transforms of in-situ IR data using a convolutional neural network (CNN) to predict location-dependent mechanical properties. Using data from twelve additively manufactured thin walls from a DLD process where an IR camera monitored the entire build, the researchers demonstrated that the developed CNN could produce predictions with less than 5% variations on the measured ultimate tensile strength (UTS[3]) values despite challenges from the uncertainty present in the experimental data.

[3]The UTS is the maximum stress that a material can withstand while being stretched or pulled before breaking.

In Fang et al. [23] researchers similarly developed a 1D CNN based on thermal histories to predict location-dependent mechanical properties but used experimentally validated, finite element simulations of a DLD process to produce the thermal histories. The researchers noted that an advantage to using finite element models to generate the thermal history enabled the CNN to be trained on error free data and enabled the data-driven CNN to predict and monitor mechanical properties for AM builds with more complex geometries. However, it was noted that for computational efficiency the finite element model neglected some powder-scale details. The model's prediction had an $R^2 = 0.67$ for UTS on the test set.

2.2.3 Thermal History Models using Melt Pool Thermal Images

In Guo et al. [34] researchers adopted a Spatial-Temporal Conditional AutoRegressive (STCAR) model with an AR(1,1) variant to characterize and forecast thermal history. The developed model offered a one time step ahead forecast of the melt pool. In addition to this, a hierarchical two-level control chart system was implemented to identify anomalous behavior. The researchers validated their methodology using simulation data as well as a case study using single-track thin wall construction using Ti-6AL-4V powder. The researchers summarized the HAZs by partitioning the data into equally sized *data trunks* (intervals) then producing trunk-wise sample means so as to reduce the dimensionality of the data. In the numerical study, sample data was generated for 10×10 images. The models produced in the numerical study were effective in identifying anomalies with a lower false positive level compared to related works. In the case study, the reduced dimensionality of the data was not reported. The researchers reported two goodness-of-fit statistics, the Deviance Information Criterion (DIC) and the Log pseudo marginal likelihood (LPML), indicating the ther-

mal forecasting performance of their model was better than another STCAR alternative, but no error metrics related to the temperature prediction were provided.

Khanzadeh et al. [45] took a different approach by developing a model to forecast thermal image streams in a layer-wise manner. Researchers developed a tensor network-based regression model to forecast one-layer ahead behavior. Autoregressive-1 (AR(1)) tensor-on-tensor regression was used where only the preceding layer was used to forecast the consequent layer to avoid high computational costs. Additionally, images were constrained to areas of interest (AOIs), 30×30 crops of the original 752×480 images. Higher order partial least squares (HOPLS) was used to estimate the parameters of the model. The researchers tested the methodology on a 75 layer, double-track thin wall construction using Ti-6AL-4V powder in a DLD process as well as simulation results. On the experimental data, researchers achieved an average root mean square error of prediction (RMSEP) of 30.250°C per image with a minimum training time of 766.85 seconds and a minimum testing time of 343.58 seconds.

CHAPTER III

METHODOLOGY

Various techniques were used to construct a forecasting model of the pyrometer sensor data, and to adapt the model to new data. The final model is an ensemble of models constructed using decomposed components of a tensor representation of the data. The focus of this chapter is on the methodology used to perform the decomposition, the final ensemble structure, the techniques used to model the decomposed components, and the algorithms used to update the decomposition structure to new data.

Here bold, lower case notation is adopted to indicate vectors. Bold, upper case notation indicates matrices. Unless otherwise noted, italics for upper case and lowercase indicates a scalar. All analysis and modeling was done using the R programming language [69].

3.1 Tensor Rank Decomposition

The sensor data under consideration is a collection of images captured at uniform time intervals during the printing process. This data collection process naturally lends itself in viewing the dataset as a *tensor* since images have consistent dimensions and images are collected at each layer. A tensor is a multidimensional array and provides a generalization of scalars, vectors, and matrices. That is, a scalar is a zeroth-order tensor, a vector is a first-order, matrix a second-order, etc. Utilizing tensor rank decomposition methodology, namely canonical polyadic decomposition (CPD), allows us to

reduce noise [82, 47, 36, 87] in the dataset and to exploit the data structure to capture spatiotemporal correlations [76, 53, 77, 52]. CPD factorizes a tensor into a finite sum of component rank-one tensors. Here a brief description of CPD is provided. For a more complete treatment see Kolda and Bader [46]. For example, let $\mathcal{X}$ be a third-order tensor given as $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$. Then a rank R decomposition of $\mathcal{X}$ could be given as,

$$\mathcal{X} \approx [[\boldsymbol{\lambda}; \mathbf{A}, \mathbf{B}, \mathbf{C}]] \equiv \sum_{r=1}^{R} \lambda_r \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r \tag{3.1}$$

where $\mathbf{a}_r \in \mathbb{R}^I$, $\mathbf{b}_r \in \mathbb{R}^J$, $\mathbf{c}_r \in \mathbb{R}^K$, and $\mathbf{x} \circ \mathbf{y}$ is the outer product of vectors $\mathbf{x}$ and $\mathbf{y}$. $\mathbf{a}_r, \mathbf{b}_r, \mathbf{c}_r$, are referred to as the *mode vectors*. The outer product of each set of mode vectors forms a rank-one component. The *factor matrices* $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ are the collection of vectors from each component. The columns of the factor matrices are normalized to length with the weights absorbed into the vector $\boldsymbol{\lambda} \in \mathbb{R}^R$; this is done to simplify the computation. A single element $x \in \mathcal{X}$ can be given as,

$$x_{ijk} \approx \sum_{r=1}^{R} \lambda_r a_{r_i} b_{r_j} c_{r_k} \tag{3.2}$$

for $i = 1, ..., I$, $j = 1, ..., J$, and $k = 1, ..., K$. Figure 3.1 provides an illustration of CPD on a third-order tensor.

Using CPD, a tensor can be approximated to an arbitrary amount of error as the rank increases where the error is quantified using the Frobenius norm. For a third-order tensor, the Frobenius norm can be given as,

$$||\mathcal{X} - \widehat{\mathcal{X}}||_F \equiv \sqrt{\sum_{i=1}^{I} \sum_{j=1}^{J} \sum_{k=1}^{K} (x_{ijk} - \hat{x}_{ijk})^2} \tag{3.3}$$

The *variance explained* by the decomposition is a relative measure that can be calculated as,

$$\left(1 - \frac{||\mathcal{X} - \widehat{\mathcal{X}}||_F}{||\mathcal{X}||_F}\right) \times 100\% \tag{3.4}$$

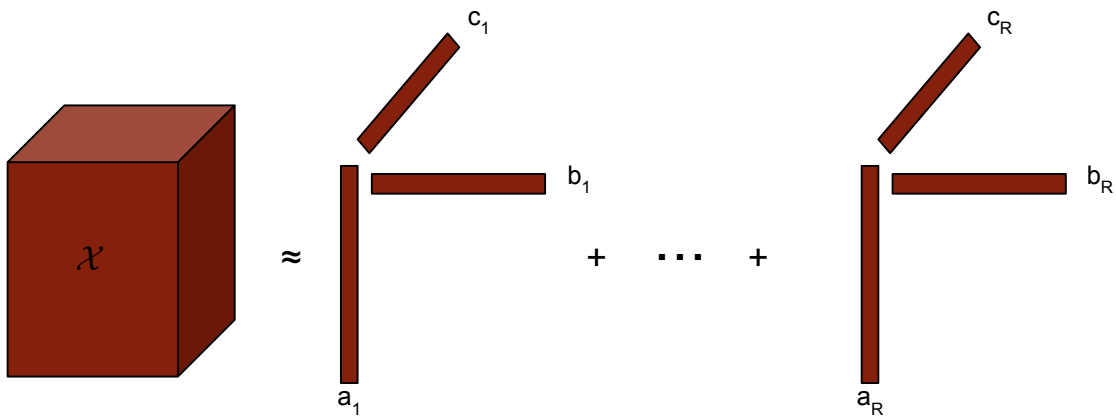

Figure 3.1

An illustration of CPD on a third-order tensor using a rank R approximation.

The rTensor [51] package is used to calculate the CPD, Frobenius norm, and variance explained. The rTensor package uses the alternating least squares (ALS) algorithm to calculate the rank decomposition.

There are a few unusual matrix operations that are pertinent to a discussion of the *CPD-ALS* algorithm: the *Kronecker product*, the *Khatri-Rao product*, the *Hadamard product*, and *matrix unfolding*. These operations are defined below as presented in Kolda and Bader.

Given $\mathbf{A} \in \mathbb{R}^{I \times J}$ and $\mathbf{B} \in \mathbb{R}^{K \times L}$, the Kronecker product of $\mathbf{A}$ and $\mathbf{B}$ is given as $\mathbf{A} \otimes \mathbf{B}$. The resulting matrix is of size $IK \times JL$ and is defined as,

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{1,1}\mathbf{B} & a_{1,2}\mathbf{B} & \cdots & a_{1,J}\mathbf{B} \\ a_{2,1}\mathbf{B} & a_{2,2}\mathbf{B} & \cdots & a_{2,J}\mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{I,1}\mathbf{B} & a_{I,2}\mathbf{B} & \cdots & a_{I,J}\mathbf{B} \end{bmatrix} \tag{3.5}$$

Given $\mathbf{A} \in \mathbb{R}^{I \times K}$ and $\mathbf{B} \in \mathbb{R}^{J \times K}$, the Khatri-Rao product, known as the *matching columnwise* Kronecker product, is given as $\mathbf{A} \odot \mathbf{B}$, and produces a matrix of size $IJ \times K$. The operation is defined as,

$$\mathbf{A} \odot \mathbf{B} = [\mathbf{a}_1 \otimes \mathbf{b}_1 \quad \mathbf{a}_2 \otimes \mathbf{b}_2 \quad \cdots \quad \mathbf{a}_K \otimes \mathbf{b}_K] \tag{3.6}$$

where $\mathbf{a}_i$ and $\mathbf{b}_i$ are the i^{th} column vectors of $\mathbf{A}$ and $\mathbf{B}$ respectively. Given $\mathbf{A} \in \mathbb{R}^{I \times J}$ and $\mathbf{B} \in \mathbb{R}^{I \times J}$, the Hadamard product is the elementwise matrix product and is given as $\mathbf{A} * \mathbf{B}$, resulting in a matrix of size $I \times J$. The operation is defined as,

$$\mathbf{A} * \mathbf{B} = \begin{bmatrix} a_{1,1}b_{1,1} & a_{1,2}b_{1,2} & \cdots & a_{1,J}b_{1,J} \\ a_{2,1}b_{2,1} & a_{2,2}b_{2,2} & \cdots & a_{2,J}b_{2,J} \\ \vdots & \vdots & \ddots & \vdots \\ a_{I,1}b_{I,1} & a_{I,2}b_{I,2} & \cdots & a_{I,J}b_{I,J} \end{bmatrix} \tag{3.7}$$

Lastly, matrix unfolding, also called *matricization*, is the process of mapping a third-order or higher tensor to a matrix. A mode-n unfolding of $\mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is denoted as $\mathbf{Y}_{(n)}$. A formal definition is given in Kolda and Bader, but a simple example which they present will provide a greater understanding. Let the $\mathcal{Y} \in \mathbb{R}^{3 \times 4 \times 2}$ with the *frontal slices* given as,

$$\mathbf{Y}_1 = \begin{bmatrix} 1 & 4 & 7 & 10 \\ 2 & 5 & 8 & 11 \\ 3 & 6 & 9 & 12 \end{bmatrix}, \quad \mathbf{Y}_2 = \begin{bmatrix} 13 & 16 & 19 & 22 \\ 14 & 17 & 20 & 23 \\ 15 & 18 & 21 & 24 \end{bmatrix}$$

Then the mode-n unfoldings are,

$$\mathbf{Y}_{(1)} = \begin{bmatrix} 1 & 4 & 7 & 10 & 13 & 16 & 19 & 22 \\ 2 & 5 & 8 & 11 & 14 & 17 & 20 & 23 \\ 3 & 6 & 9 & 12 & 15 & 18 & 21 & 24 \end{bmatrix}, \quad \mathbf{Y}_{(2)} = \begin{bmatrix} 1 & 2 & 3 & 13 & 14 & 15 \\ 4 & 5 & 6 & 16 & 17 & 18 \\ 7 & 8 & 9 & 19 & 20 & 21 \\ 10 & 11 & 12 & 22 & 23 & 24 \end{bmatrix},$$

$$\mathbf{Y}_{(3)} = \begin{bmatrix} 1 & 2 & 3 & \cdots & 10 & 11 & 12 \\ 13 & 14 & 15 & \cdots & 22 & 23 & 24 \end{bmatrix}$$

With those definitions supplied, the ALS algorithm for a third-order tensor is presented below as adapted from Kolda and Bader where $\mathbf{A}^{\dagger}$ denoted the Moore-Penrose pseudoinverse of $\mathbf{A}$. In its default version, $D \sim \mathcal{N}(0, 1)$. As will be outlined in chapter 3.3.2, previously learned modes can be used for initialization.

Algorithm 1: *CPD-ALS*$(\mathcal{X}, R, D)$

Result: $[[\lambda; \mathbf{A}, \mathbf{B}, \mathbf{C}]]$

initialize $\mathbf{A} \in \mathbb{R}^{I \times R}, \mathbf{B} \in \mathbb{R}^{J \times R}, \mathbf{C} \in \mathbb{R}^{K \times R}$ with samples drawn from D

repeat

$\mathbf{A} := \mathbf{X}_{(1)}(\mathbf{B} \odot \mathbf{C})(\mathbf{B}^{\top}\mathbf{B} * \mathbf{C}^{\top}\mathbf{C})^{\dagger}$; normalize columns of $\mathbf{A}$ and store norms in $\boldsymbol{\lambda}$

$\mathbf{B} := \mathbf{X}_{(2)}(\mathbf{A} \odot \mathbf{C})(\mathbf{A}^{\top}\mathbf{A} * \mathbf{C}^{\top}\mathbf{C})^{\dagger}$; normalize columns of $\mathbf{B}$ and store norms in $\boldsymbol{\lambda}$

$\mathbf{C} := \mathbf{X}_{(3)}(\mathbf{A} \odot \mathbf{B})(\mathbf{A}^{\top}\mathbf{A} * \mathbf{B}^{\top}\mathbf{B})^{\dagger}$; normalize columns of $\mathbf{C}$ and store norms in $\boldsymbol{\lambda}$

until *convergence or maximum iterations reached*;

For some processes, a significant amount of the variation within the data can be accounted for using a low-rank approximation. This feature of CPD is what leads to noise reduction in the

data; signals can be preserved in low-rank approximations as they account for the *explainable* variation while noise is removed as it accounts for *unexplainable* variation. Figure 3.2 provides an illustration using the single-track, thin wall dataset A. The left plot illustrates the accuracy of a low-rank approximation, with a rank 1 approximation accounting for more than 95% of the Frobenius norm. The right plot illustrates the residual distribution for $R = 1$ and $R = 7$. With both plots it is observed that accuracy increases with rank. However, gains in accuracy become more modest as the rank increases. Thus the choice of rank becomes a model selection problem.

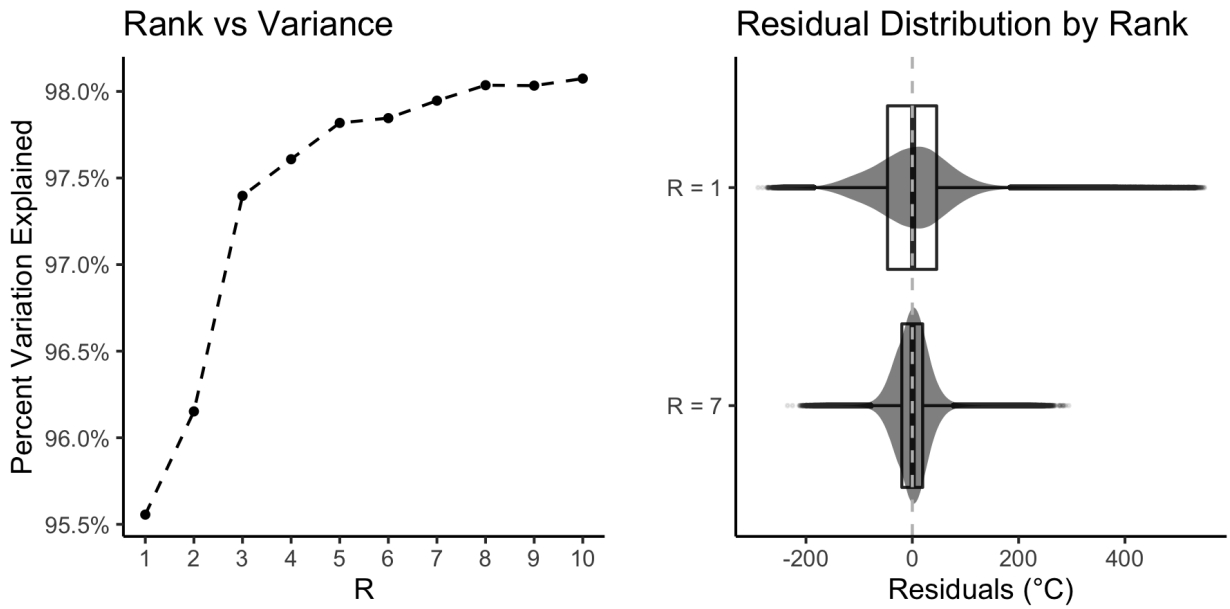

Figure 3.2

Relationship between rank and accuracy of a decomposition using the single-track, thin wall dataset A. CPD applied to a 4th order tensor $\mathcal{T} \in \mathbb{R}^{155\times155\times24\times20}$.

3.2 Ensemble Model Framework

In addition to reducing noise, the decomposition lends itself to creating simplistic, low-dimensional models that can capture spatiotemporal correlations using a separable, multiplicative framework. Individually, the mode vectors can be modeled as a function of their corresponding index. The individually constructed models are fused using the tensor rank decomposition structure. That is, an estimate $\hat{z}$ for a particular location and time could concisely be given as,

$$\hat{z} = \sum_{r=1}^{R} \lambda_r \prod_{\omega=1}^{\Omega} h_{r_\omega}(\mathbf{x}_\omega) \tag{3.8}$$

where Ω is the order of the tensor and $h_{r\omega}$ is the model for mode ω constructed as a function of the index variables of mode ω for component r, and $\mathbf{x}_\omega$ are the index variables of mode ω corresponding to a particular location and time. Figure 3.3 provides an illustration of this process.

In the illustration, an $R = 4, \Omega = 3$ formulation is used. $\mathcal{X}$ represents the original data while $\widehat{\mathcal{X}}$ is the $R = 4$ decomposition. $\mathcal{L}$ is the set of models constructed from the individual modes of each component. The form depicted is used in chapter IV for the 3D tensor representation.

In the case studies using the single-track, thin wall dataset A, images are taken at 24 unique track coordinates for each layer. For each image, x and y coordinates reference pixel locations in the image. Applying CPD with a rank R decomposition will produce R components with each component having mode vectors associated with the x pixel coordinates, y pixel coordinates, track position, and layer in the printing process. Different modeling methods were employed in the 4D tensor formulation compared to the 3D tensor formulation. In the 4D formulation, natural cubic splines are used to interpolate pixel coordinates and track position, while autoregressive integrated moving average (ARIMA) models are used to forecast layer effect. The 3D formulation again

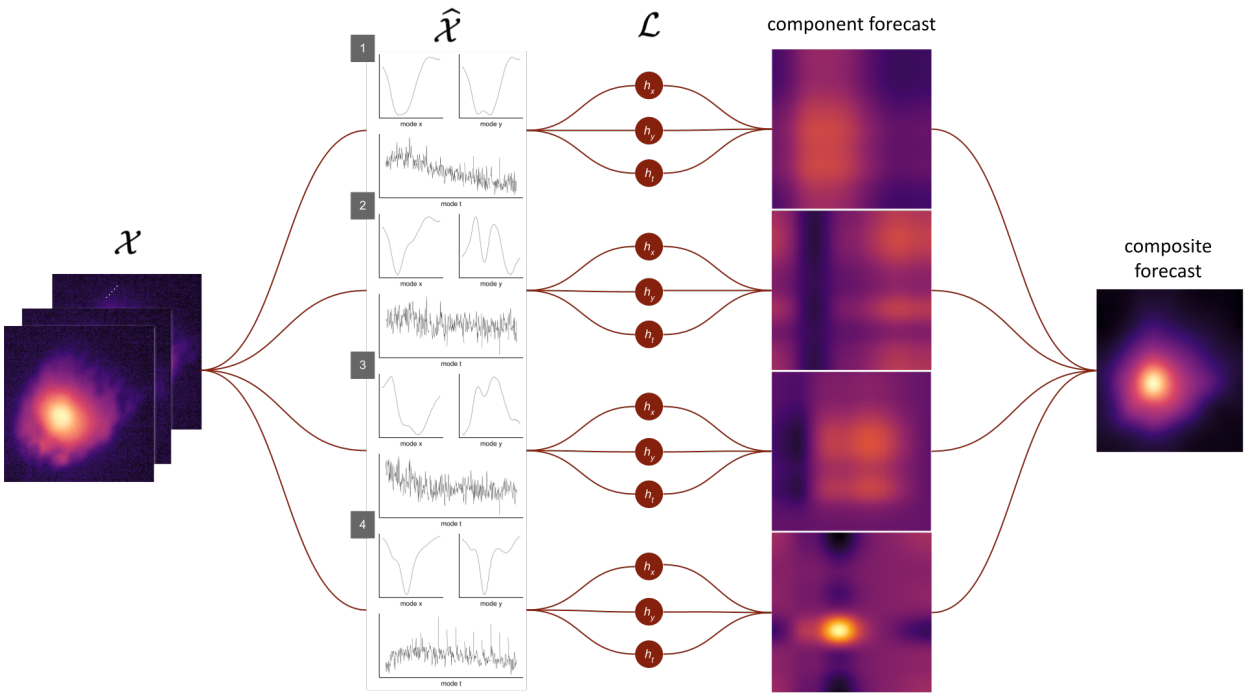

Figure 3.3

An illustration of the ensemble modeling framework.

utilized natural cubic splines to interpolate pixel coordinates. Feed-forward neural networks, with a single hidden layer, are used within an autoregressive framework to capture track-position, layer, and temporal effects. An explanation for the different models used will be given in chapter IV.

3.2.1 Natural Cubic Splines

Natural cubic splines are constructed of piecewise third-order polynomials which pass through m knots; they are both effective in modeling arbitrary functions as well as being computationally inexpensive. Consider the one-dimensional case for a set of $m+1$ points $(y_0, y_1, \ldots, y_m)$, which is

sufficient since the tensor rank decomposition reduces to one-dimensional problems. What follows is the form given in Bartels et al. [7]. Let the i^{th} section of the spline be given as,

$$Y_i(u) = a_i + b_i u + c_i u^2 + d_i u^3 \tag{3.9}$$

where $u \in [0, 1]$ and $i = 0, 1, ..., m - 1$. There are two boundary conditions, namely that,

$$\begin{aligned} Y_i(0) &= y_i = a_i \\ Y_i(1) &= y_{i+1} = a_i + b_i + c_i + d_i \end{aligned} \tag{3.10}$$

As there are four coefficients at least four equations are required to solve for a particular $Y_i(u)$. Two more equations can be obtained by adding constraints on the first derivative D_i,

$$\begin{aligned} Y_i^{(1)}(0) &= D_i = b_i \\ Y_i^{(1)}(1) &= D_{i+1} = b_i + 2c_i + 3d_i \end{aligned} \tag{3.11}$$

Thus, all four coefficients can be symbolically solved yielding,

$$\begin{aligned} a_i &= y_i \\ b_i &= D_i \\ c_i &= 3(y_{i+1} - y + i) - 2D_i - D_{i+1} \\ d_i &= 2(y_i - y_{i+1}) + D_i + D_{i+1} \end{aligned} \tag{3.12}$$

with the D_i's being specified such that the second derivatives match at the endpoints along with a few additional conditions. This yields a symmetric, tridiagonal system of $m + 1$ equations given as,

$$\begin{bmatrix} 2 & 1 & & & & & & & \\ 1 & 4 & 1 & & & & & & \\ & 1 & 4 & 1 & & & & & \\ & & . & . & . & . & & & \\ & & & & 1 & 4 & 1 & & \\ & & & & & 1 & 4 & 1 & \\ & & & & & & 1 & 2 & \end{bmatrix} \begin{bmatrix} D_0 \\ D_1 \\ D_2 \\ \vdots \\ D_{m-2} \\ D_{m-1} \\ D_m \end{bmatrix} = \begin{bmatrix} 3(y_1 - y_0) \\ 3(y_2 - y_0) \\ 3(y_3 - y_1) \\ \vdots \\ 3(y_{m-1} - y_{m-3}) \\ 3(y_m - y_{m-2}) \\ 3(y_m - y_{m-1}) \end{bmatrix} \tag{3.13}$$

The STATS [70] package is used to compute the natural splines in this study.

3.2.2 ARIMA Models

AutoRegressive Integrated Moving Average (ARIMA) models are a popular approach in time series forecasting, being effective in capturing autocorrelations in time series data by incorporating past observations (autoregression) and past forecast errors (moving average) into forecasts. What follows is the form given by Hyndman and Athanasopoulos [40]. A non-seasonal ARIMA model can be given as,

$$y'_t = c + \phi_1 y'_{t-1} + \cdots + \phi_p y'_{t-p} + \theta_1 \epsilon_{t-1} \tag{3.14}$$

where y'_t is the differenced time series, c is some constant, ϕ_i is the coefficient for the $t-i$ lagged value of y'_t, and θ_i is the coefficient for the $t-i$ lagged error of y'_t. The equation can be rewritten using backshift notation[1] as,

$$(1 - \phi_1 B - \cdots - \phi_p B^p)(1 - B)^d y_t = c + (1 + \theta_1 B + \cdots + \theta_q B^q)\epsilon_t \tag{3.15}$$

This model is labeled as an *ARIMA*(p, d, q) *model* where p is the order of the autoregressive component, d is the degree of differencing, and q is the order of the moving average component. The FORECAST [41] package is used to construct ARIMA models in this study.

3.2.3 Neural Network Autoregression

Neural Network AutoRegression (NNAR) models allow complex, nonlinear relationships between the response of interest and its predictors, being able to incorporate both lagged response values and concurrent predictors [40]. An illustration of a NNAR model used in this study is given in Figure 3.4.

To forecast a response at $y(t)$, the current layer in the printing process, the track position, and the previous p response values are used. The NNAR in this study uses a *multilayer feed-forward* architecture with a single hidden layer. The FORECAST [41] package is used to construct NNAR models in this study. The value p is determined automatically according the Akaike information criterion (AIC) of a linear autoregressive model, while the size of the hidden layer was set to half the size of the input layer plus one. A decay parameter equal to ten was used which provides a conservative model. An ensemble of 30 NNAR models were used per mode vector with forecasts

[1]The backshift operator, B, is defined as $By_t = y_{t-1}$. The exponent of B determines the number of differences, i.e., $B^d y_t = y_{t-d}$.

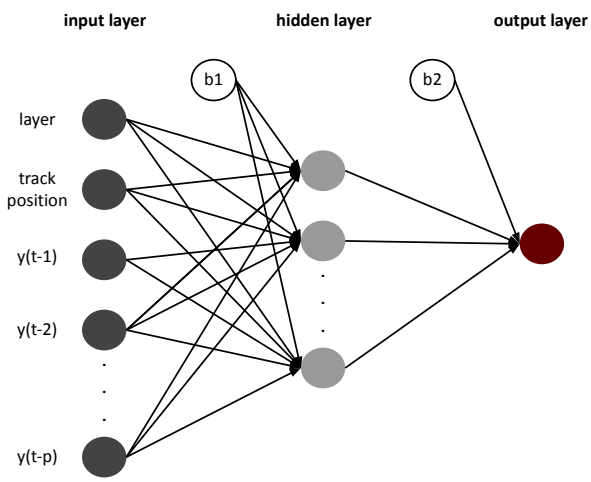

Figure 3.4

An illustration of the NNAR model used in this study.

averaged. The reader can refer to Bishop [10] for a detailed treatment on multilayer feed-forward networks.

3.3 Tensor Update Procedure

The developed tensor update procedure is composed of two systems: an *intra*-layer update and an *inter*-layer update. The intra-layer update is made as new data becomes available while the inter-layer update is made at the completion of each layer. These systems were developed to augment a tensor decomposition using a 3D framework, though they could be adjusted to account for a 4D framework as well. The following sections describe these processes in detail. Chapter IV presents

the experimental results related to this approach. As a convenience, the notation $\mathbf{A}[i_1{:}i_2, j_1{:}j_2]$ is used to indicate tensor slices. That is, given a second-order tensor $\mathbf{A} \in \mathbb{R}^{10\times10}$ where,

$$\mathbf{A} = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,10} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,10} \\ \vdots & \vdots & \ddots & \vdots \\ a_{10,1} & a_{10,2} & \cdots & a_{10,10} \end{bmatrix}, \text{ then } \mathbf{A}[3{:}5, 7{:}9] = \begin{bmatrix} a_{3,7} & a_{3,8} & a_{3,9} \\ a_{4,7} & a_{4,8} & a_{4,9} \\ a_{5,7} & a_{5,8} & a_{5,9} \end{bmatrix}$$

For brevity, an empty ":" is defined to indicate the entire index. That is, $\mathbf{A} = \mathbf{A}[:, :]$.

3.3.1 Intra-layer Update

In this procedure, the objective is to update the temporal mode of a decomposition while the spatial modes remain fixed. Let $\mathcal{T}_0 \in \mathbb{R}^{I\times J\times K}$ be a tensor of melt pool thermal images where I and J give the spatial domain and K gives the temporal domain. Let $\widehat{\mathcal{T}_0} = \sum_{r=1}^{R} \lambda_r \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r^0$ be a rank R decomposition of $\mathcal{T}_0$, where $\mathbf{a}_r = \{a_{r_1}, a_{r_2}, \ldots, a_{r_I}\}$, $\mathbf{b}_r = \{b_{r_1}, b_{r_2}, \ldots, b_{r_J}\}$, and $\mathbf{c}_r^0 = \{c_{r_1}, c_{r_2}, \ldots, c_{r_K}\}$. Let $\mathbf{X}_1 \in \mathbb{R}^{I\times J}$ be an image observed at temporal index K+1. Then, let $\mathcal{T}_1 \in \mathbb{R}^{I\times J\times K}$ where $\mathcal{T}_1[:, :, 1{:}(K\text{-}1)] = \mathcal{T}_0[:, :, 2{:}K]$ and $\mathcal{T}_1[:, :, K] = \mathbf{X}_1$. Define $\widehat{\mathcal{T}_1} = \sum_{r=1}^{R} \lambda_r \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r^1$ where $\mathbf{c}_r^1 = \{c_{r_2}, c_{r_3}, \ldots, c_{r_{K+1}}\}$. $c_{r_{K+1}}$ is solved for by minimizing the Euclidean norm between $\mathbf{X}_1$ and the reconstruction where $c_{r_{K+1}}$ is initialized at c_{r_K}. This becomes an R-dimensional optimization problem. In practice, a sample of $\mathbf{X}_1$ is compared to a sample of the reconstruction to reduce computation time. Trivially, $\mathcal{T}_n$, $\widehat{\mathcal{T}_n}$, and $\mathbf{c}_r^n$ can be recursively defined for an arbitrary n. This produces the following equation,

$$\min_{c_{r_{K+n}}\in\mathbb{R}} \left\| \mathbf{X}_n - \sum_{r=1}^{R} c_{r_{K+n}} \lambda_r \mathbf{a}_r \circ \mathbf{b}_r \right\| \tag{3.16}$$

Figure 3.5 provides an illustration of the intra-layer update. The Nelder-Mead method from the STATS [70] library is used to numerically solve equation 3.16.

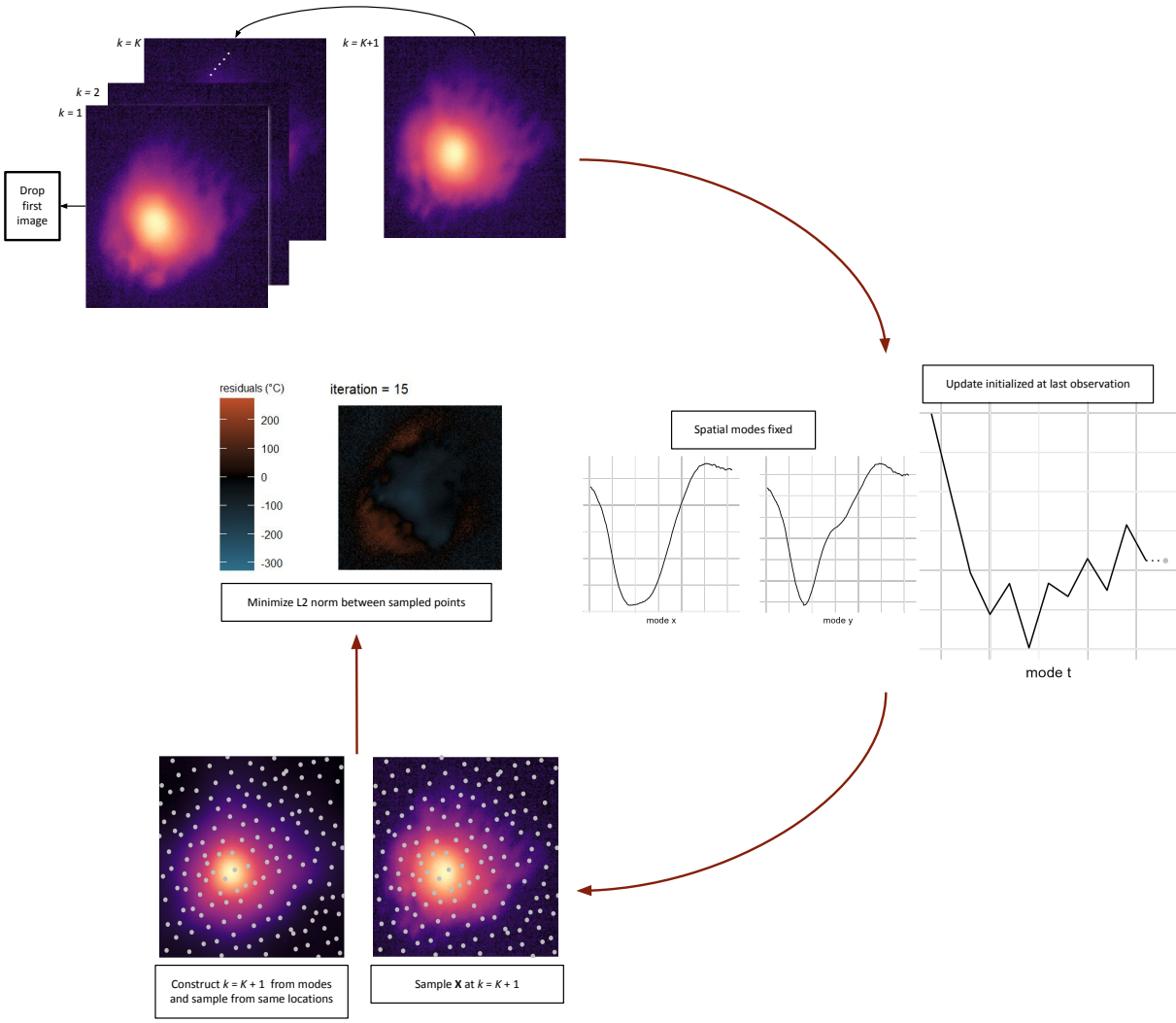

Figure 3.5

An illustration of the intra-layer update procedure.

3.3.2 Inter-layer Update

In this procedure, a combined update of the spatial and temporal modes is performed at the completion of each layer. The key to this approach is using the *CPD-ALS* algorithm, with the modes being initialized with the previously solved spatial modes and the adjusted temporal modes.

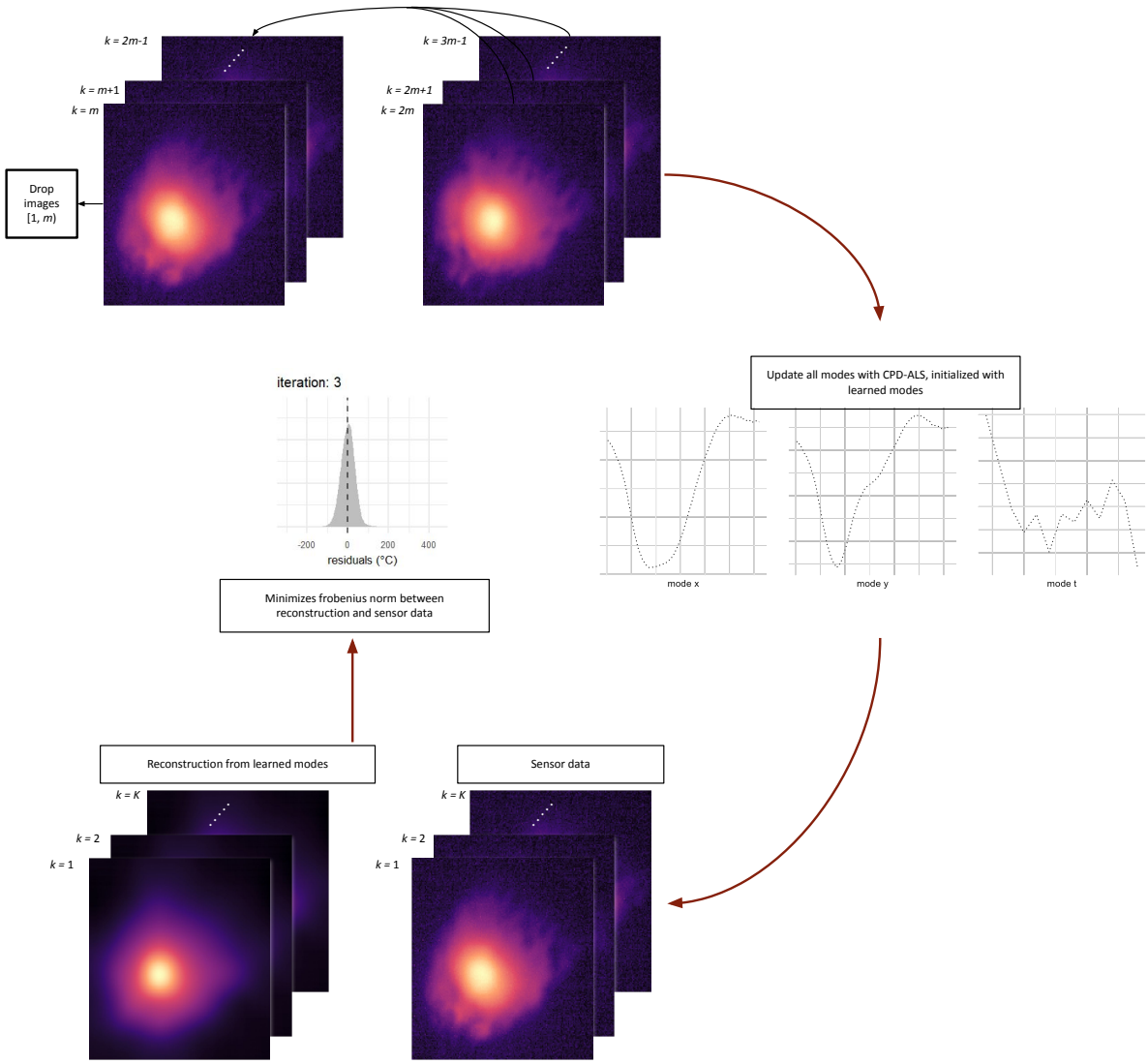

Figure 3.6

An illustration of the inter-layer update procedure.

Building off the example given in the previous section, let m be the number of images in a particular layer, where $\mathbf{c}_r^m = \{c_{r_{1+m}}, c_{r_{2+m}}, \ldots, c_{r_{K+m}}\}$. After solving for $c_{r_{K+m}}$, if the relative change between $||\mathcal{T}_m - \widehat{\mathcal{T}}_m||_F$ and $||\mathcal{T}_0 - \widehat{\mathcal{T}}_0||_F$ is greater than a given tolerance then the *CPD-ALS* algorithm is applied where the modes are initialized with there solved values with the solution being stored in $\widehat{\mathcal{T}}_m$. Otherwise, $\widehat{\mathcal{T}}_m$ is left unchanged and the procedure continues. That is, let

$$e_m = \frac{||\mathcal{T}_m - \widehat{\mathcal{T}}_m||_F - ||\mathcal{T}_0 - \widehat{\mathcal{T}}_0||_F}{||\mathcal{T}_m||_F} \tag{3.17}$$

If $e_m > \delta$ then $\widehat{\mathcal{T}}_m^* = CPD\text{-}ALS(\mathcal{T}_m, R, [[\boldsymbol{\lambda}; \mathbf{A}, \mathbf{B}, \mathbf{C}^m]])$. Otherwise, $\widehat{\mathcal{T}}_m^* = \widehat{\mathcal{T}}_m$. In this study, $\delta = 10^{-5}$. Figure 3.6 gives an illustration of this procedure.

3.4 Transfer Learning Framework

The transfer learning framework uses the update procedure so that a learned decomposition can be quickly adjusted to data from a new process. Transfer learning has lately received much attention in AM modeling research as it enables quickly learning new processes where limited data may be available [19, 39, 65, 24, 80]. Following the notation given in section 3.3, let $\mathcal{T}_0 \in \mathbb{R}^{I \times J \times K}$ be a tensor of melt pool thermal images from a dataset labeled A and $\widehat{\mathcal{T}}_0 = \sum_{r=1}^{R} \lambda_r \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r^0$ be a rank R decomposition of $\mathcal{T}_0$. Let $\mathbf{Y}_k \in \mathbb{R}^{I \times J}$ be an image observed at temporal index $k = 0, 1, 2, ..., K$ for a new dataset labeled B. Let $\widehat{\mathcal{B}}_k \in \mathbb{R}^{I \times J \times K}$ be an approximation of dataset B where $\widehat{\mathcal{B}}_0 = \widehat{\mathcal{T}}_0$. For each k within a particular layer, the intra-layer update is applied. Different from the intra-layer update, however, the temporal mode $\mathbf{c}_r^1 = \{c_{r_1^*}, c_{r_2}, \ldots, c_{r_K}\}$ where $c_{r_1^*}$ is initialized at c_{r_1} and solved via equation 3.18,

$$\min_{c_{r_k^*} \in \mathbb{R}} \left\| \mathbf{Y}_k - \sum_{r=1}^{R} c_{r_k^*} \lambda_r \mathbf{a}_r \circ \mathbf{b}_r \right\| \tag{3.18}$$

As each image, $\mathbf{Y}_k$, is observed, images are stored in a new tensor $\mathcal{B}_k \in \mathbb{R}^{I \times J \times K}$ where $\mathcal{B}_k[:,:,1{:}k] = \mathbf{Y}_{1,2,\ldots,k}$ and $\mathcal{B}_k[:,:,k{+}1{:}K] = \widehat{\mathcal{B}}_k[,,k{+}1{:}K]$. In practice, the values of $\mathbf{c}_r^k$ indexed greater than k are mean centered by the mean of values less than or equal to k to provide continuity in the reconstruction. That is,

$$\mathbf{c}_r^k[k{+}1{:}K] := \mathbf{c}_r^k[k{+}1{:}K] - \overline{\mathbf{c}_r^k[k{+}1{:}K]} + \overline{\mathbf{c}_r^k[1{:}k]}$$

where the wide bar indicates the mean.

At the end of any particular layer, the inter-layer update is applied. Using similar notation as given in section 3.3.2, let m be the number of images in a particular layer, where here $\mathbf{c}_r^m = \{c_{r_1}, c_{r_2}, \ldots, c_{r_m}, \ldots, c_{r_K}\}$. After solving for c_{r_m}, if the relative change between $||\mathcal{B}_m - \widehat{\mathcal{B}}_m||_F$ and $||\mathcal{B}_0 - \widehat{\mathcal{B}}_0||_F$ is greater than a given tolerance then the *CPD-ALS* algorithm is applied where the modes are initialized with there solved values with the solution being stored in $\widehat{\mathcal{B}}_m$. Otherwise, $\widehat{\mathcal{B}}_m$ is left unchanged and the procedure continues. That is, let

$$e_m = \left| \frac{||\mathcal{B}_m - \widehat{\mathcal{B}}_m||_F - ||\mathcal{B}_0 - \widehat{\mathcal{B}}_0||_F}{||\mathcal{B}_m||_F} \right| \tag{3.19}$$

If $e_m > \delta$ then $\widehat{\mathcal{B}}_m^* = CPD\text{-}ALS(\mathcal{B}_m, R, [[\boldsymbol{\lambda}; \mathbf{A}, \mathbf{B}, \mathbf{C}^m]])$. Otherwise, $\widehat{\mathcal{B}}_m^* = \widehat{\mathcal{B}}_m$. Figure 3.7 gives an illustration of this procedure.

3.5 Error Metrics

Three metrics are used to assess the accuracy of the prediction models: the Root Mean Squared Error (RMSE), the Normalized RMSE (NRMSE), and the Mean Absolute Percentage Error (MAPE). The RMSE is given as,

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{N}} \tag{3.20}$$

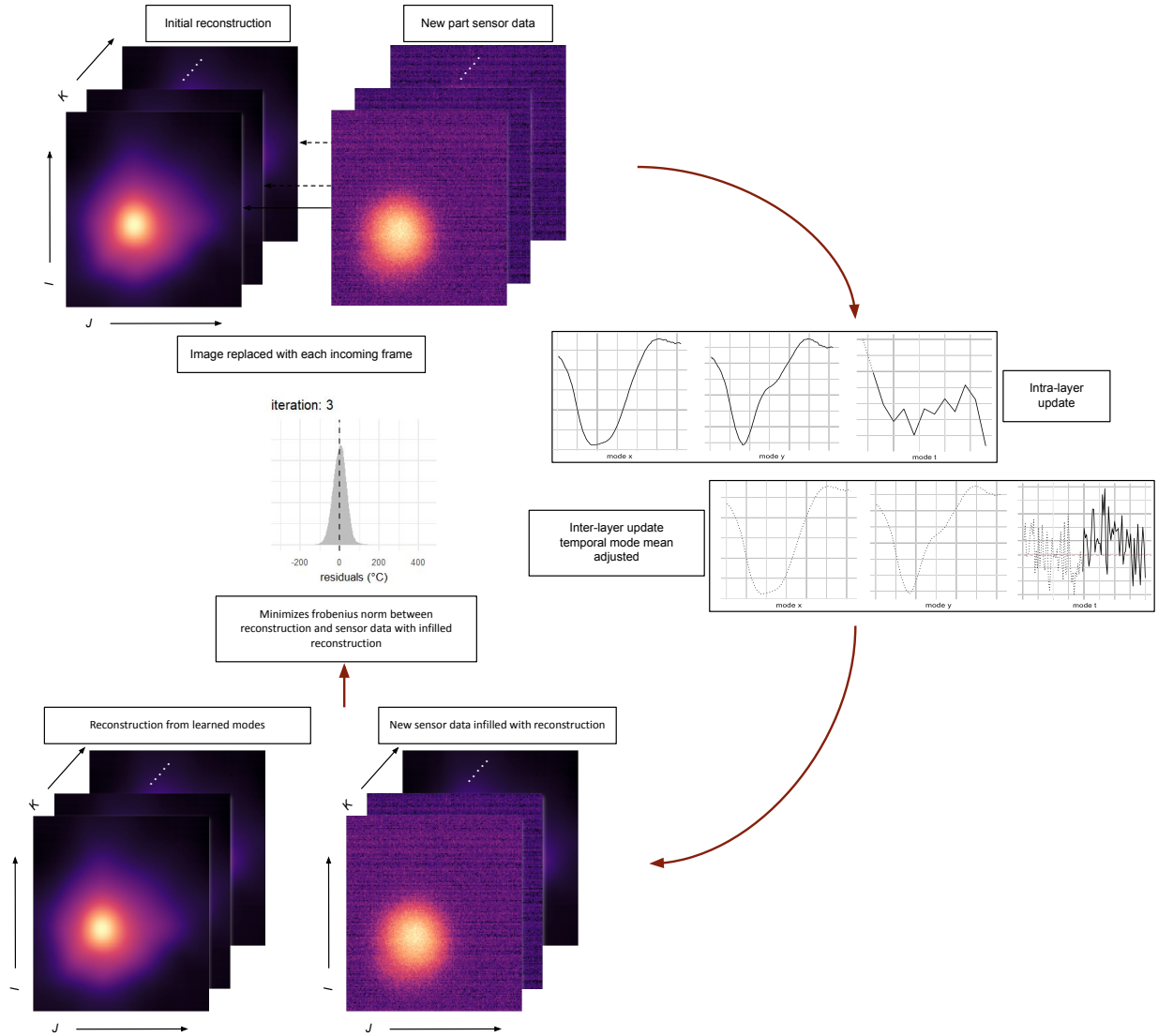

Figure 3.7

An illustration of the transfer learning framework.

where y_i is the observed value, $\hat{y}_i$ is the predicted value, and N is the number of (observed, predicted) pairs. The NRMSE is given as,

$$\text{NRMSE} = \frac{1}{y_{max} - y_{min}} \sqrt{\frac{\sum_{i=1}^{N} (y_i - \hat{y}_i)^2}{N}} \times 100\% \tag{3.21}$$

where y_{max} and y_{min} give the maximum and minimum of the observed values, respectively. Lastly, the MAPE is given as,

$$\text{MAPE} = \frac{1}{N} \sum_{i=1}^{N} \left| \frac{y_i - \hat{y}_i}{y_i} \right| \times 100\% \tag{3.22}$$

The RMSE indicates the deviation of predicted values from the observed value. The NRMSE provides a scaled version of the RMSE, giving a relative measurement of error which allows a comparison between models of different datasets. MAPE also provides a relative measurement of error, and a simpler interpretation than RMSE and NRMSE, as it is the scaled first moment of the absolute value of the residuals whereas RMSE and NRMSE provide the square root of the second moment.

CHAPTER IV

RESULTS

4.1 Data Description and Preprocessing

An OPTOMEC LENS^{TM} 750 equipped with a 1 kW Nd:YAG laser (IPG) was used to additively manufacture a thin wall structures from Ti-6Al-4V powder. A dual-wavelength pyrometer (Stratonics, Inc) was affixed above the OPTOMEC LENS^{TM} 750 machine to capture melt pools while an IR camera (Sierra-Olympic Technologies, Inc. Viento320) was used to capture global heat behavior. Each image captured by the pyrometer produced files containing a 480 x 752 matrix where the value of each element corresponds to a pixel of the pyrometer camera. Units of the values are in degrees Celsius, but the specific measurements were not corrected for the emissivity of the Ti-6Al-4V during its manufacture. The pyrometer sensor had a temperature range of $1,000 - 2,500$°C. A naming convention of the files was used that provided important information regarding the AM process. This naming convention indicated the relative time (seconds) the image was captured, the relative x, y, and z coordinates (millimeters), and the layer in the AM process. Specific details about the experimental setup can be found in Marshall *et al.* [55]. In this study, only the data produced by the pyrometer was used.

Datasets from two thin wall builds are used in this study. These datasets are referred to hereafter as tw_A and tw_B. Table 4.1 provides metadata related to the builds found within the datasets. $\mathbf{N}_i$, $\mathbf{N}_l$, and $\mathbf{N}_y$ give the number of CSV files, layers printed, and unique y coordinates,

Table 4.1

Metadata for the thin wall pyrometer datasets.

	$\mathbf{N}_i$	$\mathbf{N}_l$	$\mathbf{N}_y$	**Part Dimensions** (L × W × H)	**Time**$_{total}$ (s)	**Time**$_{layer}$ (s)
tw_A	2,007	79	27	52.62 mm × 1.78 mm × 39.78 mm	443.954	5.620
tw_B	1,564	60	34	63.89 mm × 1.78 mm × 30.09 mm	457	7.617

respectively. **Time**$_{total}$ and **Time**$_{layer}$ provide the total time of the build and the average time per layer, respectively. tw_B is directly referenced in Marshall [55].

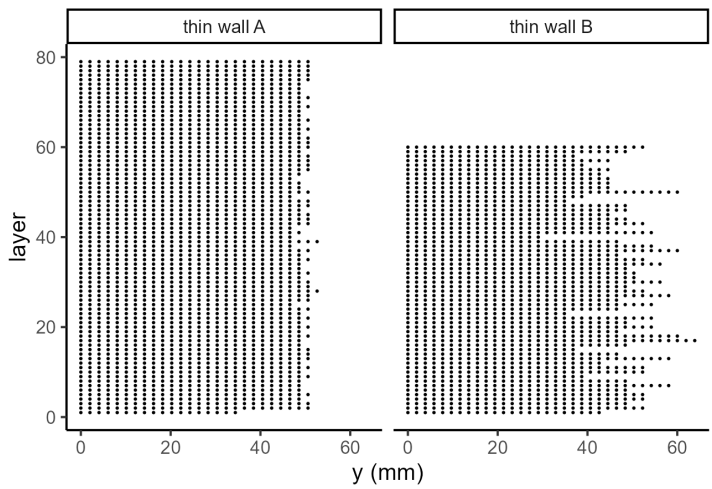

Figure 4.1

Locations of captured images for tw_A and tw_B.

While tw_A contained 79 layers with 27 unique y-coordinate locations, approximately 6% of the images were *missing*, that is, samples at particular y-coordinates were not present for all layers. All of the missing images occurred at the first layer and at the last three y-coordinates. To simplify the evaluation of the methodology in the initial experiments, the first layer and the last three y-

coordinates were dropped. tw_B had significantly more missing images, with approximately 23% of the images missing. No limitations were imposed on tw_B, as the impetus for a more robust model formulation was partly due to handling missing data. Figure 4.1 illustrates the locations of the captured thermal images for tw_A and tw_B. Each point represents a captured image where the x-axis gives the y-coordinate location and the y-axis gives the layer. Additionally, in both tw_A and tw_B each image contained some missing data denoted as zeros in the CSV files. This missing data constituted a minor portion of each image, though tw_B had roughly an order of magnitude more of missing data. Table 4.2 provides a description of the proportion of missing data per image for both tw_A and tw_B.

Table 4.2

Proportion of missing data per image.

	Minimum	**1st Quartile**	**Median**	**Mean**	**3rd Quartile**	**Maximum**
tw_A	0.012%	0.039%	0.042%	0.042%	0.045%	0.070%
tw_B	0.103%	0.385%	0.406%	0.372%	0.423%	0.526%

For model training purposes, missing data was replaced by calculating the mean value of the adjacent, non-zero pixels. In evaluating the performance of the model, the raw data was used, excluding pixels with zero values.

Although the resolution of each image was 752×480, it was determined that the relevant information captured by the pyrometer sensor, i.e. the melt pool and HAZs, were contained within a small subset of the image. Thus, each image was cropped from 752×480 to 155×155 where

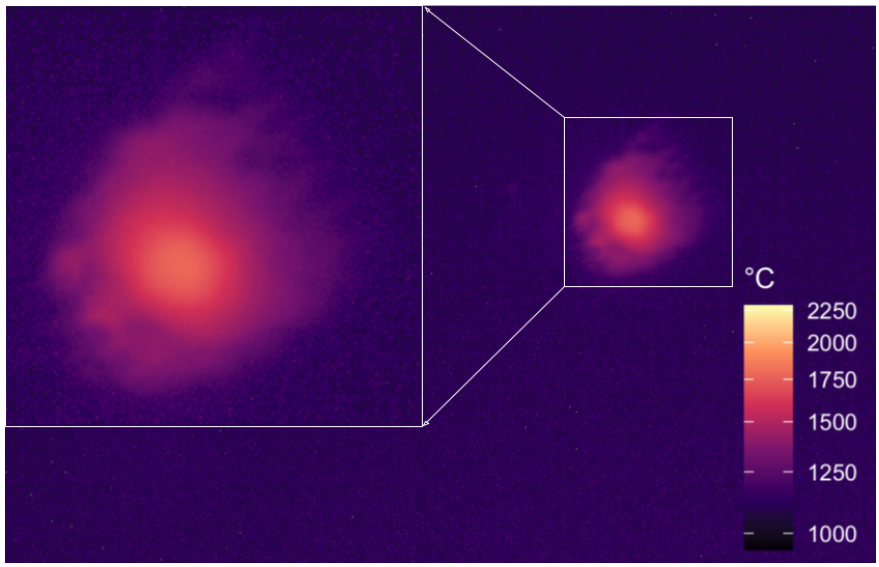

Figure 4.2

Illustration of the cropped portion of the raw pyrometer sensor image. The provided image depicts the first observation of the second layer. Color is $\log_{10}$ scaled to highlight differences.

the crop was centered by identifying the pixel with the highest temperature for each image. Figure 4.2 provides an illustration of the cropped image.

4.2 4D Tensor Framework Experiment

In this experiment, a 4D tensor representation of tw_A is used to represent spatial behavior, single layer behavior, and multilayer behavior. A rank decomposition is applied to the tensor, reducing dimensionality of the data and noise within the data, enabling the construction of low-cost, one-dimensional models. The ensemble of these models enables accurate, multilayer forecasts. Figure 4.3 gives an example of the 4D tensor structure used in this experiment. As discussed in Chapter III, cubic splines were used to model the x and y coordinates and the track position, while an ARIMA model was used to model layer effect.

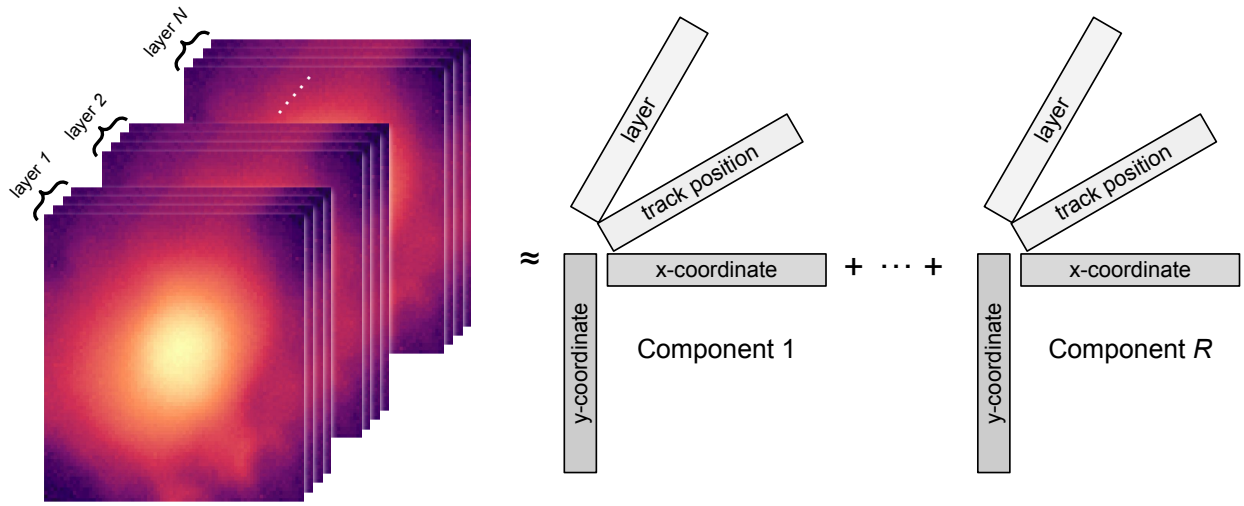

Figure 4.3

An illustration of the 4D tensor structure and decomposition.

The proposed methodology is compared to a seasonal naïve model, demonstrating that meaningful trends are identified using the proposed method. However, the 4D representation requires a rigid data structure which imposes several limitations. The advantages and disadvantages of this methodology will be discussed at the conclusion of this section.

4.2.1 Evaluation Procedure

To determine the viability of the proposed method, the tensor model was compared with a *seasonal naïve model*. The seasonal naïve model uses the value that corresponds to the last observed value from the same season to make forecasts, that is, it uses the values observed at the previous layer at the same location as its forecast. Seasonal naïve models are effective when the data contains seasonal behavior (similar patterns per layer) and future movements are unpredictable [40]. This model is established as a baseline to challenge whether layer-wise trends are identifiable within the data. The thesis is that if the tensor model can produce more accurate forecasts than the

seasonal naïve model, then this provides evidence that the method successfully identifies trends in the printing process. This will serve as a proof of concept of the methodology.

Additionally, of concern in this evaluation was the optimal rank and *memory*, i.e. the number of layers used in the tensor formulation, of the tensor model. Initial exploration of number of layers, N, and rank, R, demonstrated decreasing forecast accuracy with $N > 40$ and $R > 10$, thus the scope of the tests were limited to $N \leq 40$ and $R \leq 10$. To determine these parameters, time series cross-validation (TSCV) was applied. A tensor would be formulated as $\mathcal{T} \in \mathbb{R}^{155\times155\times24\times N}$ where $N \in \{2, 3, \ldots, 40\}$ and would be decomposed using $R = 1, 2, \ldots, 10$. The resulting tensor could be labeled as the training tensor. For each combination of N and R, a model would be trained then tested on the subsequent five layers. After each test, training layers would shift by one layer as would testing layers. This process would continue 30 times resulting in distributions of residuals obtained for each testing set.

Layers were selected so that models were evaluated on the same testing sets. That is, when $N = 40$, the first training tensor would consist of layers 2 through 41, with the testing layers being 42 through 46. For $N = 39$, the first training tensor would consist of layers 3 through 41, with the testing layers again being 42 through 46. Lastly, with $N = 2$, the first training tensor would consist of layers 40 and 41, with the testing layers the same as before. Figure 4.4 provides illustrations of the TSCV tests.

The initial TSCV test is used to select the optimal rank and memory of the tensor model using residuals on a *per layer* basis. The second TSCV test compares the selected tensor model to the seasonal naïve model using residuals on a *per image* basis, providing a more granular assessment.

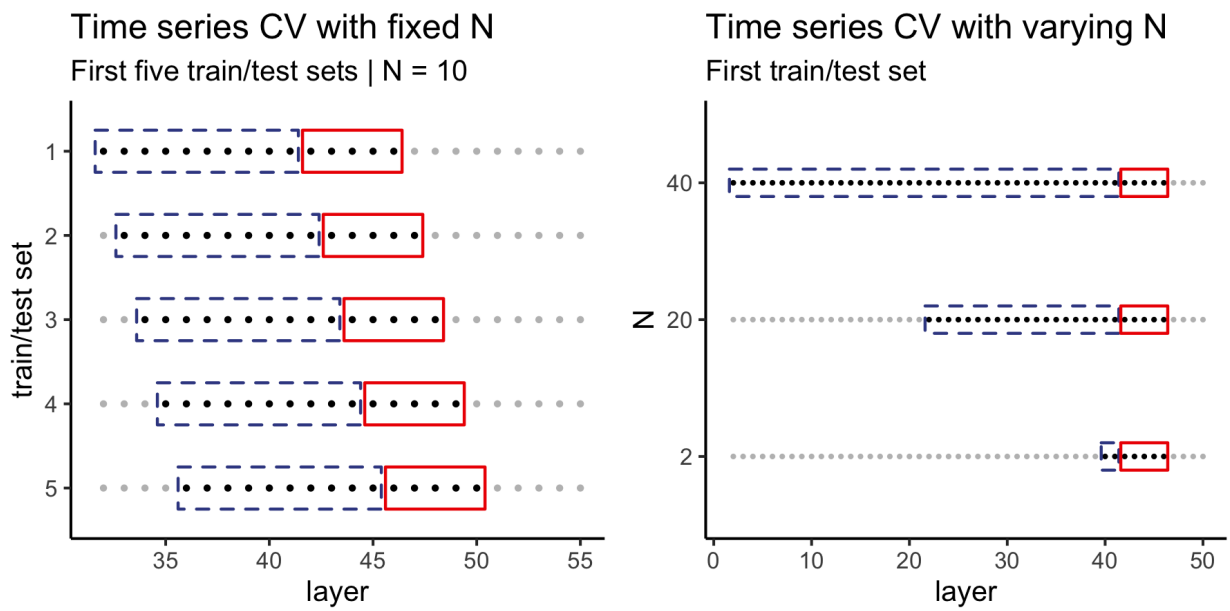

Figure 4.4

The left figure illustrates TSCV with fixed N for the first five train/test sets, where the y-axis identifies the train/test set. The right figure illustrates TSCV with varying N for the first train/test set, where the y-axis identifies the value of N. The blue, dashed box indicates the train set while the red, solid box indicates the test set. Each point represents a layer. Gray points represent unused layers.

4.2.2 TSCV Results and Comparison to Seasonal Naïve Model

Using the residuals collected during TSCV, the RMSE was calculated for each testing set, for each unique R, N, and number of layers forecasted ahead. With 30 testing sets per (R, N) combination, and 5 layers forecasted ahead, this produced a distribution of 150 RMSE values per (R, N) combination. Figure 4.5 illustrates the mean and standard deviation of the RMSE values given (R, N) combinations.

The mean and standard deviation of the RMSE are both valuable metrics. A low mean RMSE indicates that the model on the average had little deviation from the actual data it was tested on.

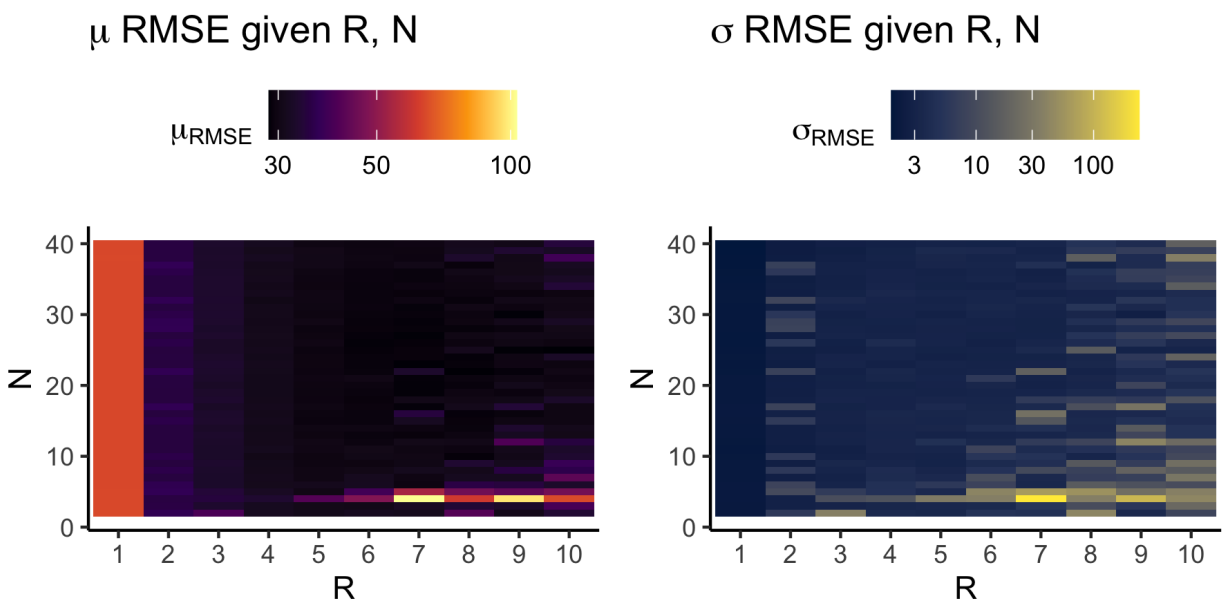

Figure 4.5

Mean and standard deviation of RMSE per (R, N) combination. Color is $\log_{10}$ scaled to highlight differences.

A low standard deviation of the RMSE indicates that the model predicted consistently regardless of where it was trained in the manufacturing cycle. Thus, both metrics were considered in model selection. The first step was to determine the selection of rank as it was responsible for explaining the majority of the variance between models.

Examination of the mean and standard deviation of the RMSE per rank was limited to $N > 5$ as a spike in error occurred with $N \leq 5$ for $R \geq 5$. Examining the Pareto frontier, ranks 1,3,4,5, and 6 all appeared on the frontier. Models constructed from ranks 3 through 6 had similar performance, being low in both metrics. Rank one models were the lowest in standard deviation, but had significantly higher error. Table 4.3 provides the mean and standard deviation for the Pareto efficient ranks. Thus consideration was limited to ranks 3 through 6.

Table 4.3

Mean and standard deviation of Pareto efficient ranks.

R	μ_{RMSE}(°C)	σ_{RMSE}(°C)
1	63.987	2.006
3	32.793	3.006
4	30.906	3.266
5	29.916	3.431
6	29.600	4.389

For the selected ranks, the Pareto set was examined for the tradeoff between mean and standard deviation of the RMSE for each predicted layer ahead. For example, for 1 layer ahead there would be 30 observations per (R, N) combination. The mean RMSE and standard deviation of the RMSE would be calculated. Then, the parameter combinations that were Pareto efficient with respect to minimizing mean RMSE and minimizing standard deviation of the RMSE would be collected. This process would be repeated for 2, 3, 4, and 5 layers ahead. In total, 21 unique combinations were Pareto efficient. It was determined to select a $(R = 5, N = 25)$ formulation as it possessed a good balance between low average error and low standard deviation of error. For brevity, the selected model is referred to as $\text{T4}_{5,25}$.

The $\text{T4}_{5,25}$ model and the seasonal naïve model were then compared using 5-layer ahead forecasts, similar to the previous test, with tests beginning at layer 27. The following plots provide a comparison between the distributions of the RMSE per image for both $\text{T4}_{5,25}$ and the naïve model.

For all five forecasted layers, $\text{T4}_{5,25}$ had an average RMSE per image of 29.701 °C with a 95% CI [29.450, 29.952], an average NRMSE per image of 3.554% with a 95% CI [3.526, 3.583], and

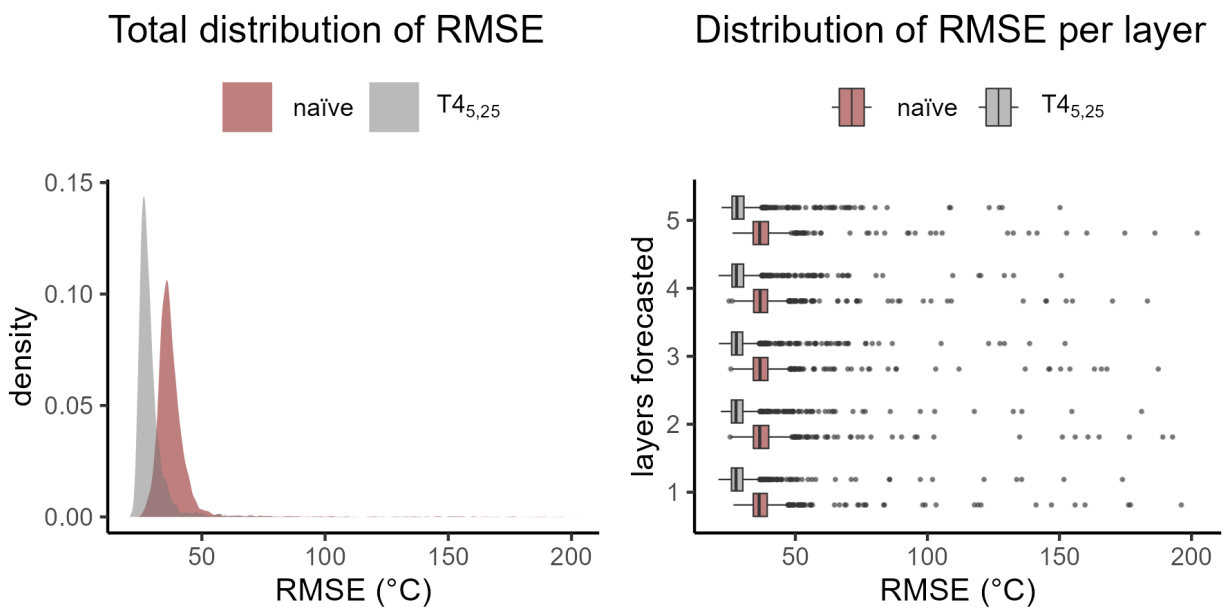

Figure 4.6

Distribution of the RMSE per image for $T4_{5,25}$ and a seasonal naïve model.

an average MAPE per image of 1.995% with a 95% CI [1.978, 2.012]. Table 4.4 provides the accuracy per image for 1-5 layers forecasted.

Table 4.4

Accuracy metrics of model $T4_{5,25}$.

Layers Ahead:	**1**	**2**	**3**	**4**	**5**
μ_{RMSE}(°C)	29.416	29.553	29.801	29.748	29.986
μ_{NRMSE}(%)	3.522	3.540	3.566	3.559	3.586
μ_{MAPE}(%)	1.971	1.984	2.005	1.999	2.017

The seasonal naïve model had an average RMSE per image of 38.454 °C with a 95% CI [38.137, 38.770], an average NRMSE per image of 4.601% with a 95% CI [4.566, 4.635], and an average

MAPE per image of 2.559% with a 95% CI [2.538, 2.581]. Table 4.5 provides the accuracy per image for 1-5 layers forecasted.

Table 4.5

Accuracy metrics of the seasonal naïve model.

Layers Ahead:	**1**	**2**	**3**	**4**	**5**
μ_{RMSE}(°C)	38.360	38.474	38.512	38.437	38.486
μ_{NRMSE}(%)	4.595	4.608	4.607	4.593	4.601
μ_{MAPE}(%)	2.553	2.561	2.566	2.556	2.561

The results provide strong evidence that $T4_{5,25}$ is more accurate than the seasonal naïve model. To further solidify these results, the two RMSE per image distributions for all five layers forecasted were compared using the Kolmogorov-Smirnov two sample test (KS test). This test compares the empirical distribution functions to check whether they are from the same distribution. Unsurprisingly, the KS test strongly indicated that cumulative distribution function (CDF) of the RMSE for $T4_{5,25}$ was greater than the CDF of the RMSE for the naïve model with a p-value $< 2.2 \times 10^{-16}$, indicating lower overall error. Thus, there can be much confidence that the model $T4_{5,25}$ provides meaningful information regarding trends within the printing process.

While the overall accuracy of model $T4_{5,25}$ is captured in the RMSE distribution plots in Figure 4.6, Figure 4.7 provides a more granular look at model performance, illustrating the model's behavior at particular observations within different layers. The average RMSE per image for a one layer forecast was 29.416 °C. The first row of Figure 4.7 provides an example of the typical performance of the model where RMSE = 26.795 °C. The second and third row of Figure 4.7

provide examples of atypical performance of the model where RMSE = 159.300 °C and RMSE = 72.906 °C, respectively. In total, 13 observations (1.105% of testing data) were more than 3 standard deviations from the mean. While anomaly detection is outside of the scope of this study, these results indicate the potential of using model forecasts or comparisons to the learned decomposition as a means of detecting anomalies. A brief treatment of the viability of detecting anomalies using the learned decomposition is given in Appendix A.

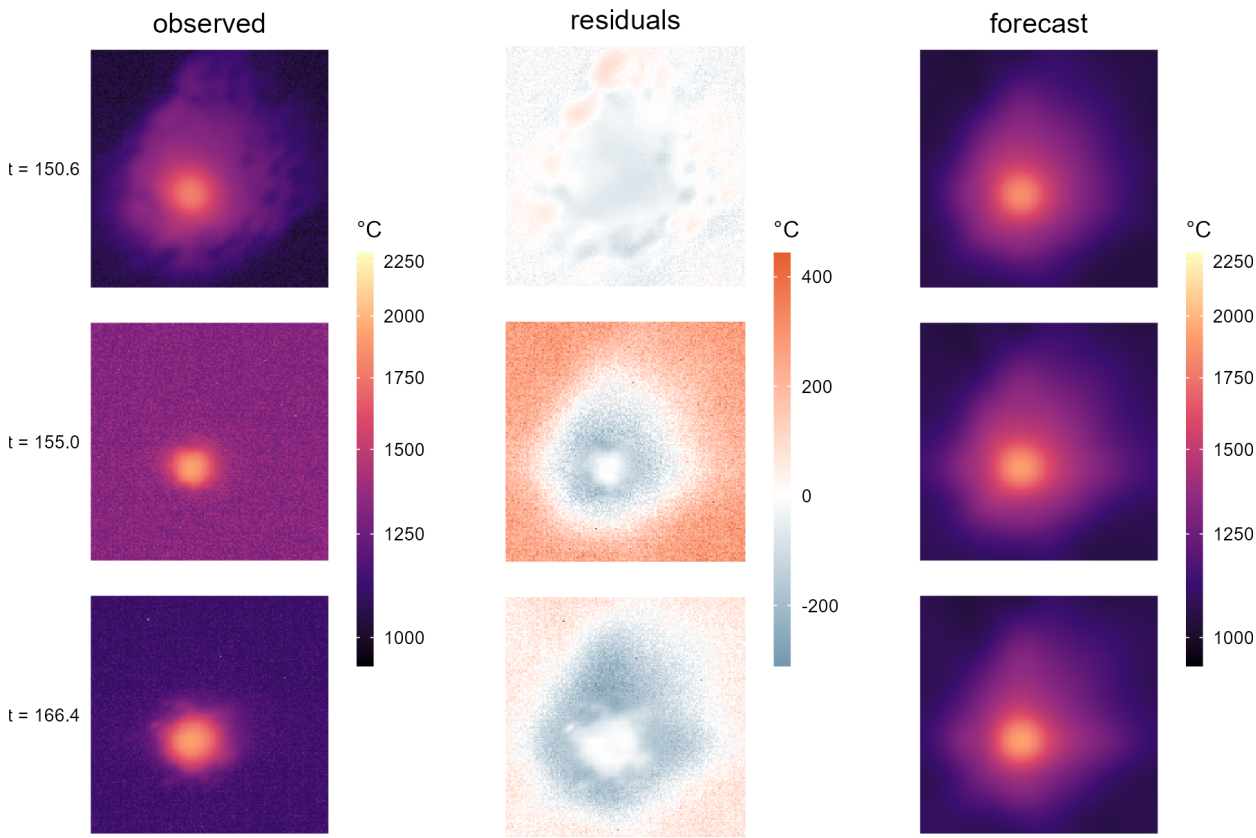

Figure 4.7

An illustration of the observed pyrometer data and a one layer ahead forecast at three different locations for $T4_{5,25}$.

Using the tensor decomposition methodology resulted in a 99.99% reduction in data size[1] while explaining 97.54% (95% CI [97.508, 97.562]) of the variance. The decomposition had a average per image accuracy[2] of 28.520 (95% CI [27.961, 29.080]). Relative to the decomposition per image RMSE, a 1 layer forecast had an average loss in accuracy of 5.123% (95% CI [4.025, 6.221]) indicating that forecasts effectively captured the information in the decomposition.

Table 4.6 provides details on the average computation time for the decomposition ($\boldsymbol{\mu}_d$), the modeling ($\boldsymbol{\mu}_m$), and the per image forecast ($\boldsymbol{\mu}_f$) along with 95% confidence intervals. Note that ($\boldsymbol{\mu}_f$) is reported in milliseconds while ($\boldsymbol{\mu}_d$) and ($\boldsymbol{\mu}_m$) are reported in seconds. All computations were done using a Dell Latitude 9420 equipped with a 3 GHz 11th Gen Intel Core i7 processor.

Table 4.6

Computation time for $T4_{5,25}$.

$\boldsymbol{\mu}_d$ (s)	$\boldsymbol{\mu}_m$ (s)	$\boldsymbol{\mu}_f$ (ms)
24.651, [20.744, 28.557]	0.214, [0.197, 0.231]	0.629, [0.592, 0.666]

4.2.3 Discussion

This experiment introduced a novel approach to forecasting the thermal history. The constructed model enabled multilayer forecasts, providing granular information of the melt pool temperature distribution. Using the tensor decomposition methodology resulted in a 99.99% reduction in data size while explaining 97.54% of the variance. The model was validated using experimental data which demonstrated high accuracy with an average NRMSE per image of 3.554%, relatively fast

[1] Training data $155 \times 155 \times 24 \times 25$ reduced to $5 \times (155 + 155 + 24 + 25)$.

[2] Per image accuracy captured using decomposition on layer 27-51.

data approximation (24.651 seconds), fast model training (0.214 seconds), and fast model forecasts (0.629 milliseconds per image).

While the 4D model succeeded in many respects, a major shortcoming was identified. The 4D representation required consistent spatial samples across time. This requirement is what led to a limitation in the data used in the experiment. A more complex geometry for a build would not have such a structure. This challenge could be addressed in a couple of ways: either applying a decomposition algorithm that can account for missing observations as seen in Acar et al. [58] or more recently in Yamaguchi and Hayahsi [91], or by restructuring the data into a 3D tensor, condensing track position and layer into a single dimension. The difficulty with the first would be the possibility of a significant number of missing observations; it could be possible that each spatial observation would be unique. A challenge with the second would be the need for a more complex model to account for the interaction of more variables. After consideration, it was determined to explore a 3D formulation of the data and compare to the 4D results.

4.3 3D Tensor Framework Experiment

In this experiment, a 3D tensor representation of tw_A is used to represent spatial behavior, single layer behavior, and multilayer behavior. Again, a rank decomposition is applied to the tensor, reducing dimensionality of the data and noise within the data, enabling the construction of low-cost, low-dimensional models. The ensemble of these models enables accurate, multilayer forecasts. Figure 4.8 gives an example of the 3D tensor structure used in this experiment. As discussed in Chapter III, cubic splines were used to model the x and y coordinate, while a NNAR

model was used to model the temporal effect. The more complex NNAR was selected over the ARIMA to account for the interaction between lagged values, layer, and track position.

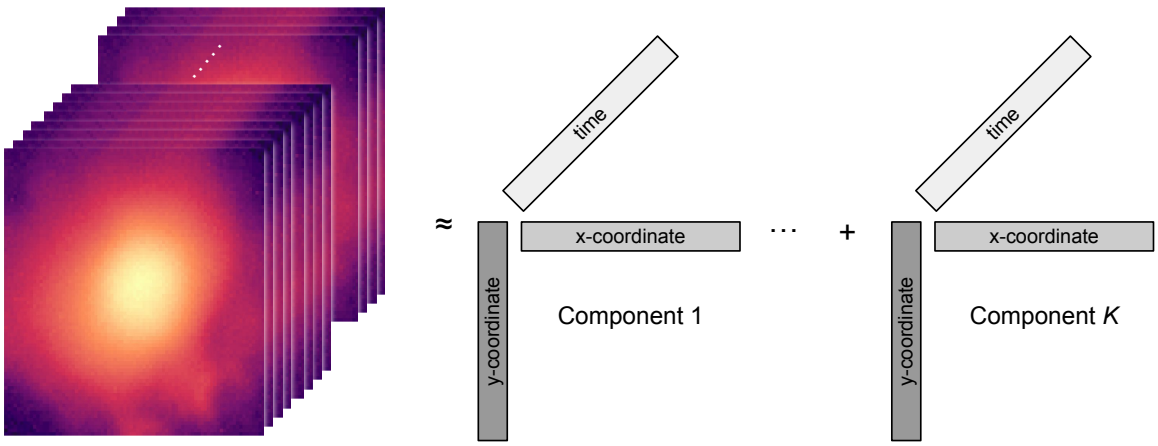

Figure 4.8

An illustration of the 3D tensor structure and decomposition.

The proposed methodology is compared to $T4_{5,25}$. The advantages and disadvantages of this methodology will be discussed at the conclusion of this section.

4.3.1 Evaluation Procedure

Similar to Section 4.2.1, TSCV was used initially to identify hyperparamters for the decomposition, then a more complete evaluation using TSCV was used to quantify the model accuracy. In this experiment, the rank of the decomposition was fixed to four as an exploration of the parameter indicated that a rank four approximation of the 3D representation provided a similar accuracy as a rank five decomposition of the 4D representation given in Section 4.2. Thus the only hyperparameter of concern in the initial evaluation using TSCV was the optimal memory of the model.

The initial TSCV procedure was performed as follows. A tensor would be formulated as $\mathcal{T} \in \mathbb{R}^{155\times155\times24N}$ where $N \in \{10, 11, \ldots, 30\}$. The resulting tensor could be labeled as the training tensor. For each N, a model would be trained then tested on the subsequent five layers. After each test, training layers would shift by one layer as would testing layers. This process would continue 30 times resulting in distributions of residuals obtained for each testing set. As in Section 4.2.1, layers were selected so that models were evaluated on the same testing sets. Again, the mean and standard deviation of the RMSE were used for model selection.

4.3.2 TSCV Results and Comparison to $\mathbf{T4}_{5,25}$

Figure 4.9 provides the distribution of RMSE per layer given N resulting from the experiment.

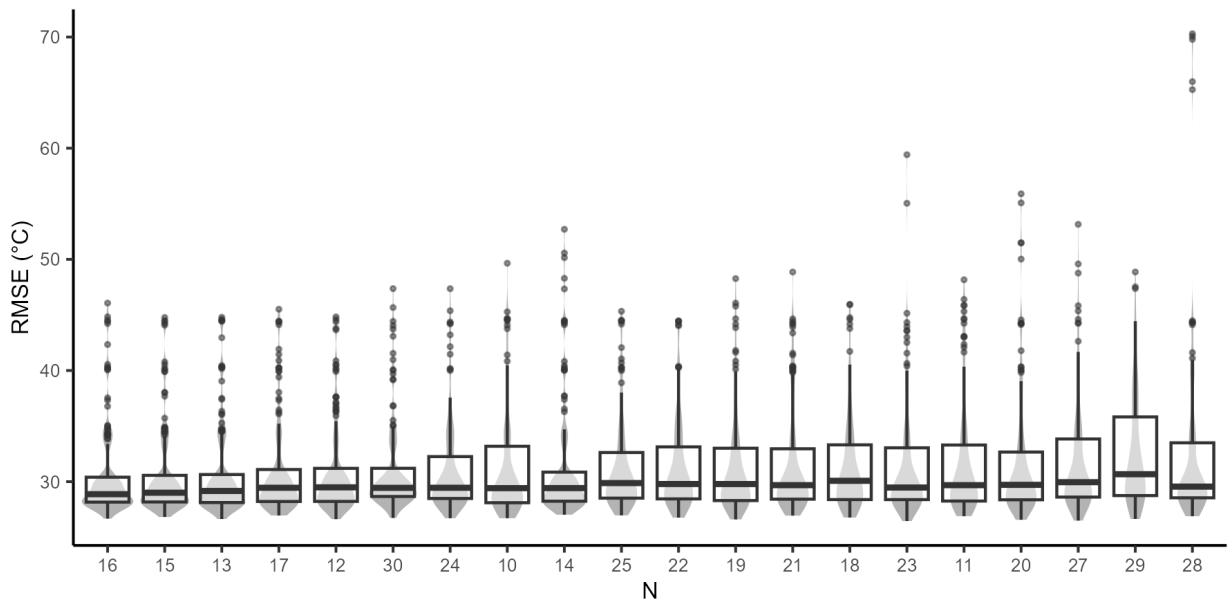

Figure 4.9

Distribution of RMSE per layer given N. Arranged by mean RMSE.

$N = 16$ produced the most accurate model with an average RMSE of 30.548 °C and a standard deviation of the RMSE of 4.096 °C. $N = 15$ was nearly as accurate and slightly more consistent with an average RMSE of 30.570 °C and a standard deviation of the RMSE of 4.041 °C. However, all N provided similar results except $N = 26$ which had a few significant outliers. $N = 26$ was excluded from the Figure 4.9 to assist in readability.

$N = 15$ was selected as fewer layers is desirable, reducing the size of the training tensor. For brevity, this model will be labeled as $T3_{4,15}$. The $T3_{4,15}$ model and the $T4_{5,25}$ model were then compared using 5-layer ahead forecasts, with comparisons beginning at layer 27 extending through 74 so that results are aligned. The following plots provide a comparison between the distributions of the RMSE per image for both $T3_{4,15}$ and $T4_{5,25}$.

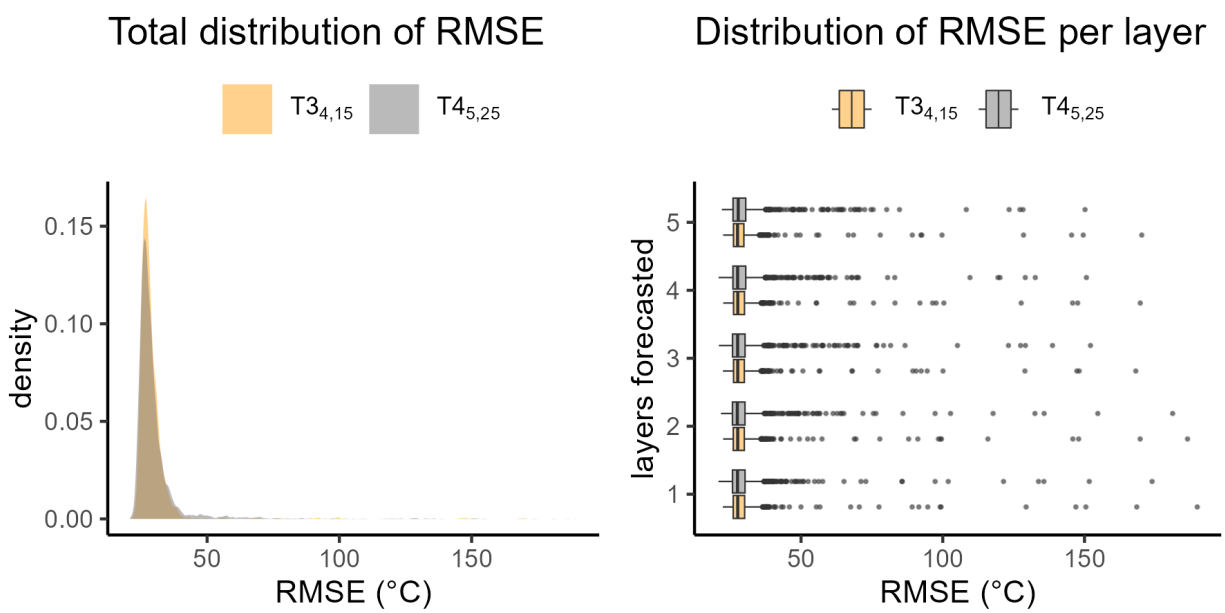

Figure 4.10

Distribution of the RMSE per image for $T3_{4,15}$ and $T4_{5,25}$.

For all five forecasted layers, $T3_{4,15}$ had an average RMSE per image of 29.323 °C with a 95% CI [29.052, 29.593], an average NRMSE per image of 3.481% with a 95% CI [3.451, 3.511], and an average MAPE per image of 1.959% with a 95% CI [1.942, 1.976]. This makes $T3_{4,15}$ slightly more accurate overall than $T4_{5,25}$ though there is overlap in the confidence intervals. Table 4.7 provides the accuracy per image for 1-5 layers forecasted.

Table 4.7

Accuracy metrics of model $T3_{4,15}$.

Layers Ahead:	**1**	**2**	**3**	**4**	**5**
$\mu_{\text{RMSE}}(^{\circ}\text{C})$	29.473	29.432	29.276	29.273	29.158
$\mu_{\text{NRMSE}}(\%)$	3.500	3.496	3.474	3.474	3.461
$\mu_{\text{MAPE}}(\%)$	1.969	1.966	1.956	1.956	1.947

By comparing Table 4.7 to Table 4.4 it can be seen that $T4_{5,25}$ is more accurate in the first forecast layer while $T3_{4,15}$ is more accurate in forecasting layers 2-5, though all values are close. Examining Figure 4.10 closely reveals that outliers for $T3_{4,15}$ are generally closer to the median for each layer forecasted, however, $T3_{4,15}$ also has the largest outliers per layer forecasted. Applying the KS test for the total RMSE distributions of $T3_{4,15}$ and $T4_{5,25}$ is inconclusive, indicating that $T3_{4,15}$ and $T4_{5,25}$ may have the same CDF.

Some unusual behavior that is exhibited in Table 4.7 is an increase in accuracy as the layers forecasted ahead increases. This is likely due to differences in some of the layers forecasted. That is, layer 1 forecasts are not made on identical temporal indices as layer 5 forecasts. Indeed, if layer 1 forecasts are aligned with layer 5 forecasts and the relative difference between the forecasts is

calculated, which is given as $\frac{\text{RMSE}_1 - \text{RMSE}_5}{\text{RMSE}_1} \times 100\%$ then accuracy *declines* by an average of 0.585% from layer 1 to layer 5. Figure 4.11 provides an illustration of this over the entire testing period where the red dashed line indicates the mean.

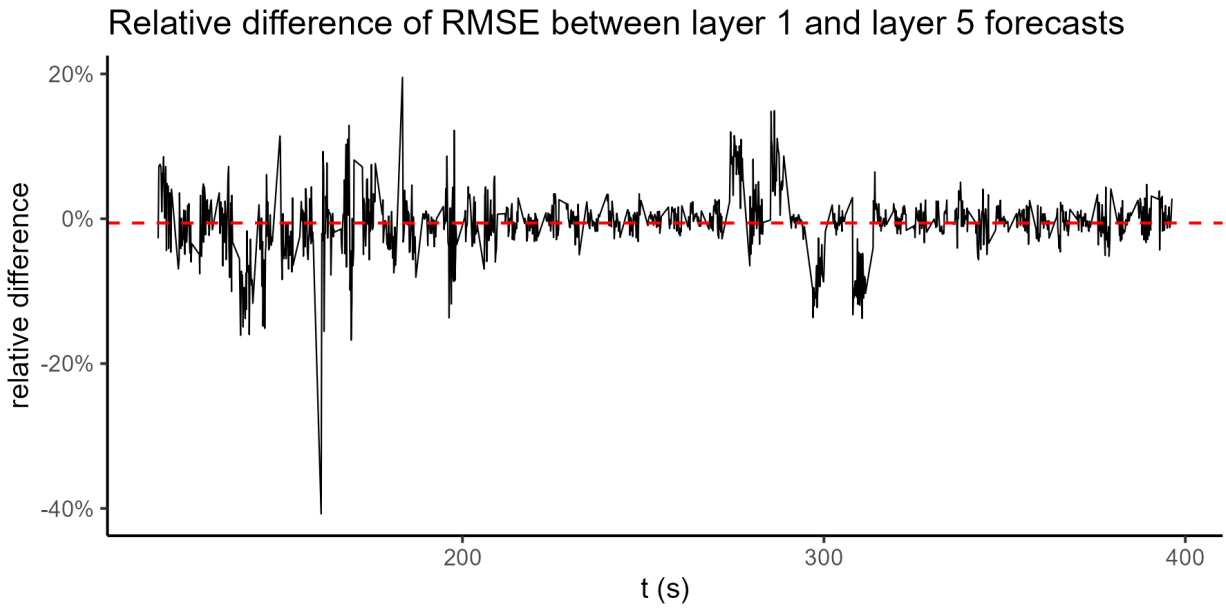

Figure 4.11

Relative difference in RMSE for a layer 1 forecast vs a layer 5 forecast for $T3_{4,15}$.

Figure 4.12 provides a granular look at model performance while giving a direct comparison to the results presented in Figure 4.7. The first row of Figure 4.12 has an RMSE = 25.473 °C (typical performance) while the second and third row of Figure 4.12 have an RMSE = 166.480 °C and RMSE = 75.719 °C, respectively (atypical performance). In comparison to the results in Figure 4.7, $T3_{4,15}$ has lower error for the instance of typical performance while $T4_{5,25}$ has lower error for atypical performance.

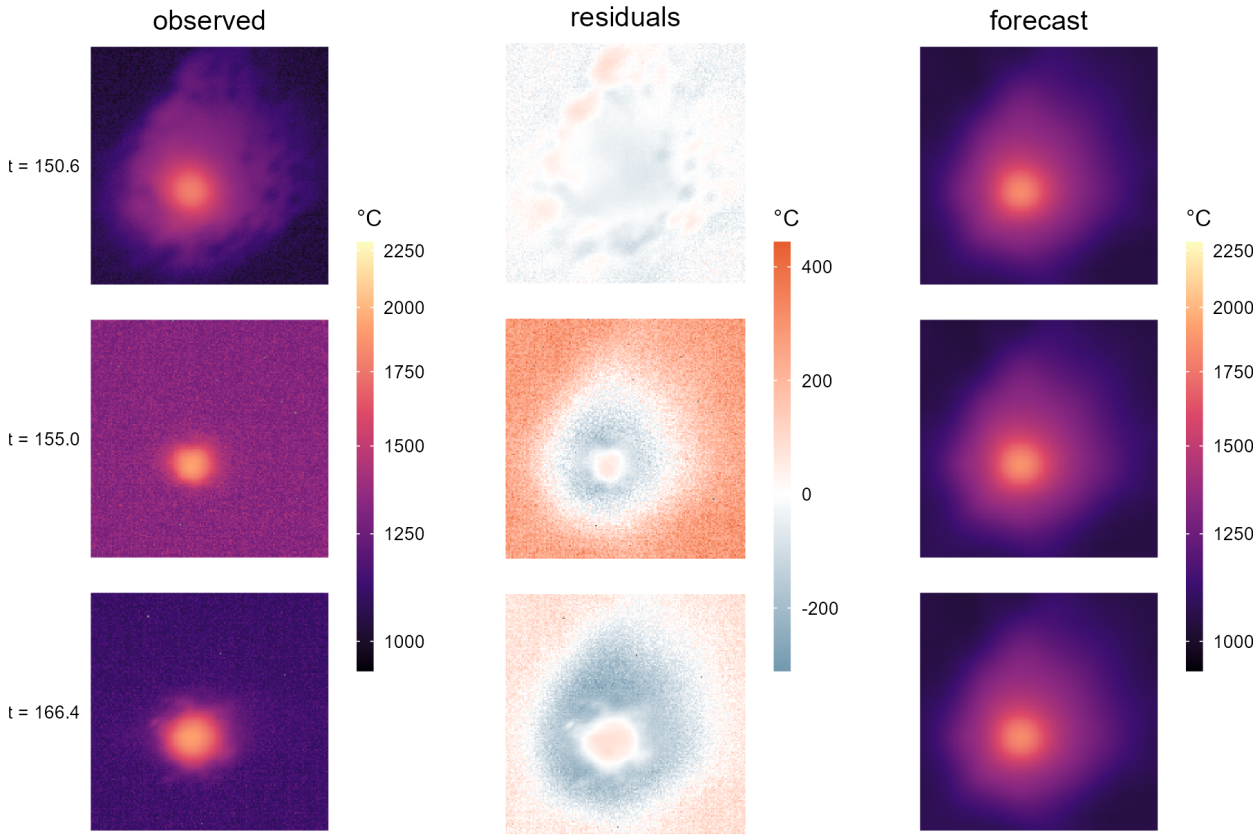

Figure 4.12

An illustration of the observed pyrometer data and a one layer ahead forecast at three different locations for $T3_{4,15}$.

Using the 3D tensor decomposition methodology resulted in a 99.97% reduction in data size[3] while explaining 97.72% (95% CI [97.698, 97.750]) of the variance. The decomposition had a average per image accuracy[4] of 26.133 (95% CI [25.711, 26.554]). Relative to the decomposition per image RMSE, a 1 layer forecast had an average loss in accuracy of 13.498% (95% CI [11.150, 15.846]) indicating that forecasts captured the information in the decomposition with a moderate

[3]Training data $155 \times 155 \times 360$ reduced to $4 \times (155 + 155 + 360)$.

[4]Per image accuracy captured using decomposition on layer 27-41.

level of effectiveness. The discrepancy between the forecast accuracy relative to the decomposition between the 4D and 3D models can be accounted for by the greater accuracy of the 3D decomposition.

Table 4.8 provides details on the average computation time for the decomposition ($\boldsymbol{\mu}_d$), the modeling ($\boldsymbol{\mu}_m$), and the per image forecast ($\boldsymbol{\mu}_f$) along with 95% confidence intervals. Note that ($\boldsymbol{\mu}_f$) is reported in milliseconds while ($\boldsymbol{\mu}_d$) and ($\boldsymbol{\mu}_m$) are reported in seconds. As with section 4.2, all computations were done using a Dell Latitude 9420 equipped with a 3 GHz 11th Gen Intel Core i7 processor.

Table 4.8

Computation time for $T3_{4,15}$.

$\boldsymbol{\mu}_d$ (s)	$\boldsymbol{\mu}_m$ (s)	$\boldsymbol{\mu}_f$ (ms)
20.657, [18.139, 23.176]	10.670, [8.540, 12.801]	9.772, [9.054, 10.490]

4.3.3 Discussion

This experiment introduced a 3D tensor structure alternative to the 4D structure given in section 4.2, for forecasting the thermal history. As with the 4D structure, the 3D tensor model enabled multilayer forecasts, providing granular information of the melt pool temperature distribution. Using the tensor decomposition methodology resulted in a 99.97% reduction in data size while explaining 97.72% of the variance. The model was validated using experimental data which demonstrated higher accuracy with an average NRMSE per image of 3.481%, faster data approximation (20.657

seconds), slower model training (10.670 seconds), and slower model forecasts (9.772 milliseconds per image).

The 3D structure theoretically overcomes the deficiency of the 4D structure as there is no need for consistent spatial samples. This idea will be tested experimentally in a later section. Furthermore, the 3D structure achieved slightly more accurate results using less data and a lower rank decomposition. However, this came at the expense of a more complex temporal model that increased the model training time and forecast time. The model training time and the decomposition time can be significantly shortened through the implementation of an update procedure to the decomposition. The experiments conducted in the previous section and this section applied full decompositions at each layer. As these decompositions are not unique, and are influenced by initialization values, models are required to be fully retrained at each layer. If consistency can be applied in the update of the decompositions, then limited retraining should be necessary. This topic, and the speed and accuracy of the decomposition update, is explored in the next section.

4.4 Update Procedure Experiment

In this experiment, $T3_{4,15}$ is once again used to model tw_A. However, the focus of this experiment is to evaluate the update procedure outlined in section 3.3. A number of tasks were identified for this experiment. Firstly, as was mentioned in section 3.3.1, a sampling strategy for the intra-layer update needed to be selected. Secondly, the accuracy and the computation time of the update procedure would need to be compared to a new decomposition. Lastly, the feasibility of augmenting the NNAR with the updated temporal mode would need to be examined. These topics are explored in this section.

4.4.1 Identifying Sampling Strategy for Temporal Update

As outlined in section 3.3.1, the update in the temporal mode is solved via equation 3.16. To simplify the optimization problem, a sample of $\mathbf{X}_n \in \mathbb{R}^{155\times155}$ and the reconstruction is used. Four sampling strategies with varying sample sizes were tested and compared: uniform random sampling, Gaussian sampling, latin hypercube sampling (LHS), and maximin LHS. Uniform random sampling assigns equal probability to each index. Gaussian sampling was constructed by calculating the density from 1 to 155 with $\mu = 77.5$ and $\sigma = 25.53475$, then performing weighted sampling. The mean and standard deviation were selected to center the distribution on the melt pool with good coverage of the HAZs[5]. LHS stratifies the sampling space in a 2-dimensional grid, where the sample is considered a LHS if and only if there is only one sample taken from each row and each column. Maximin LHS adds a constraint that maximizes the minimum pair-wise distance between points. The STATS [70] package is used to perform uniform and Gaussian sampling while the LHS [13] package is used to perform LHS and maximin LHS. Figure 4.13 provides an illustration of the sampling strategies using a sample size of 58.

The comparison between the four sampling strategies was performed as follows: using tw_A, layers 2 through 16 were used to form the training tensor and CPD was applied with a rank four decomposition. Data for layer 17 was collected to be used to quantify error in the temporal estimate. A parameter matrix, $\mathbf{P}_1 \in \mathbb{R}^{28{,}800\times4}$, was constructed that included every combination of four variables:

1. *sample size:* a vector with 10 values ranging from 10 to 155.
2. *method:* a vector identifying the four sampling methods.
3. *iterations:* a vector of length 30 providing unique identifiers for random seeds.

[5]Standard deviation was selected by optimizing on a specific image.

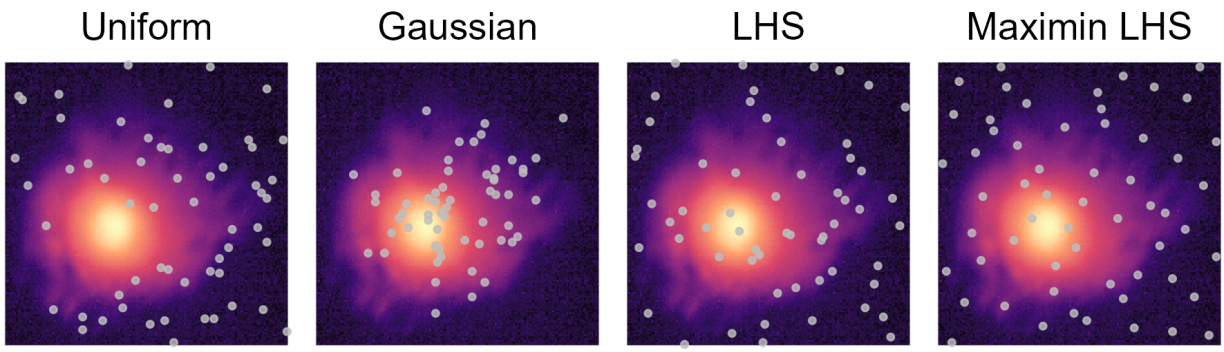

Figure 4.13

An illustration of the sampling strategies explored for the intra-layer update procedure.

4. *image key:* an identifier for the image in layer 17 to be used for the test.

An additional parameter matrix, $\mathbf{P}_2 \in \mathbb{R}^{720\times4}$, was constructed and appended to $\mathbf{P}_1$ that had a single sample size of $155^2 = 24,025$ (the entire image) with a single method labeled *all*; this was to provide a comparison in the computational cost and accuracy between using the entire image and a sample of the image. In total, 29,520 parameter combinations were evaluated.

For each parameter combination, the time to perform the optimization was collected and the error between the reconstruction and layer 17 for a given image key was collected. For all methods with sample sizes less than or equal to 155, the optimization time was similar with a median time of 0.035 seconds. Using a sample size of 24,025 produced an order of magnitude greater computation time with a median time of 0.342 seconds, explaining why differences in computation time over the sample sizes explored could not be distinguished. Regarding accuracy, a trend of diminishing return as the sample size increased was clear. Overall, maximin LHS produced the lowest error at a

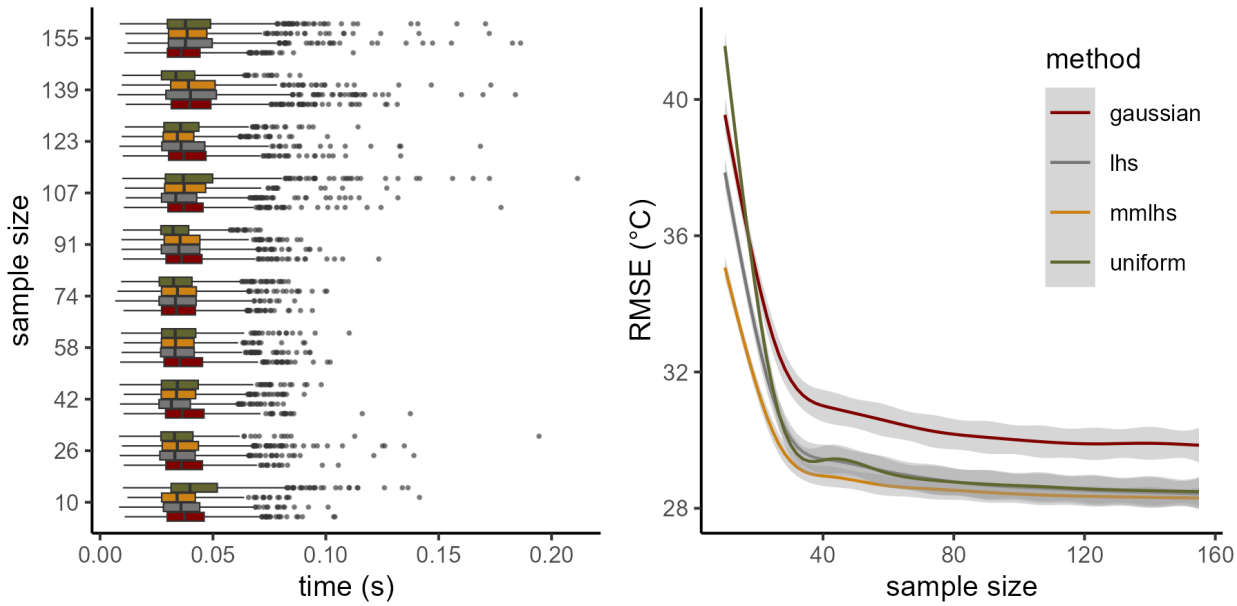

Figure 4.14

An illustration of the computation time required and accuracy for the temporal update per sample size and method.

sample size of 155 with a median RMSE of 27.542 °C. However, both LHS and uniform produced similar results at a sample size of 155 with a median RMSE of 27.663 °C and 27.688 °C. The Gaussian method had notably higher error at a sample size of 155 with a median RMSE of 28.605 °C. Figure 4.14 provides an illustration of the computation time and accuracy per sample size and method. Generalized additive models with a 95% confidence interval are used to represent error to highlight trends. As there was little trade-off for computation time between methods and for different sample sizes, maximin LHS with a sample size of 155 was selected.

Using the entire image produced no variation in the optimization procedure with the Nelder-Mead method, and the error was always lower than that of the sampling procedure. However, the results were close compared to the selected sampling procedure with a median relative difference

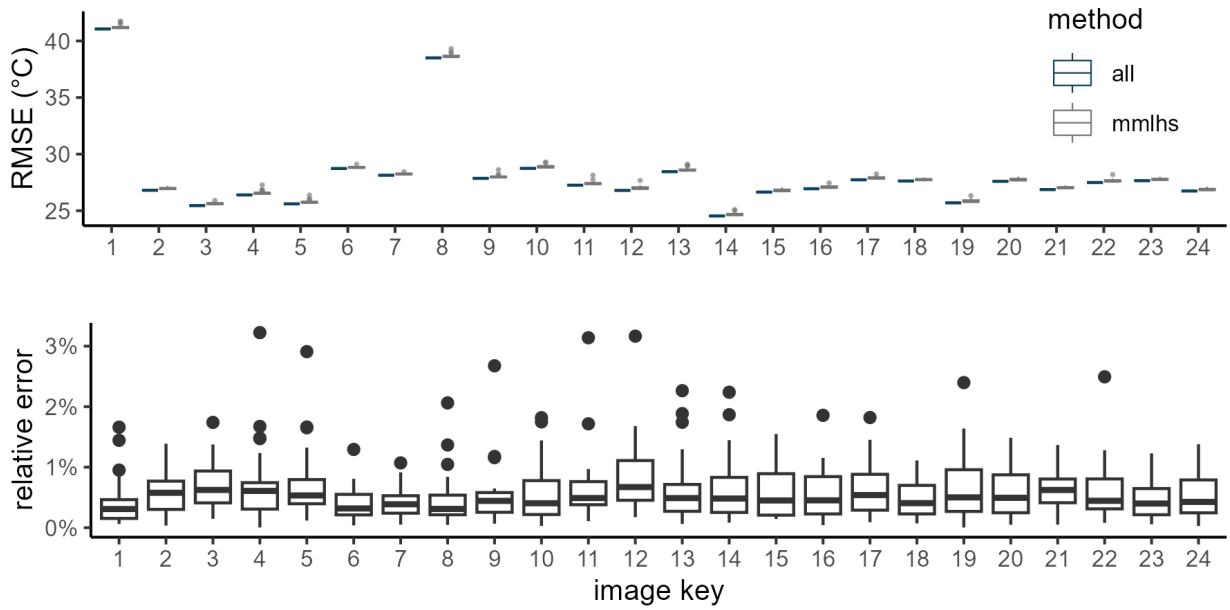

Figure 4.15

An illustration comparing the difference in accuracy between using maximin LHS with a sample size of 155 and the entire image for layer 17.

of 0.465%. Figure 4.15 provides an illustration comparing the difference in accuracy between using maximin LHS with a sample size of 155 and the entire image. The top graph provides the RMSE per image key for layer 17. As the differences are nearly indistinguishable, the bottom graph provides the relative difference in error. That is, $\frac{error_{all}-error_{mmlhs}}{error_{all}} \times 100\%$ where the error collected is the frobenius norm.

A single seed for sample generation was used in the subsequent experiments for consistency and reproducibility of results. The seed was selected by performing 10,000 iterations of a maximin LHS sample and identifying the sample structure with the lowest mean relative difference for layer 17. The sample structure with the lowest mean relative error had a random seed of 6,548 and an average relative error of 0.280%. Figure 4.16 illustrates the selection procedure and the

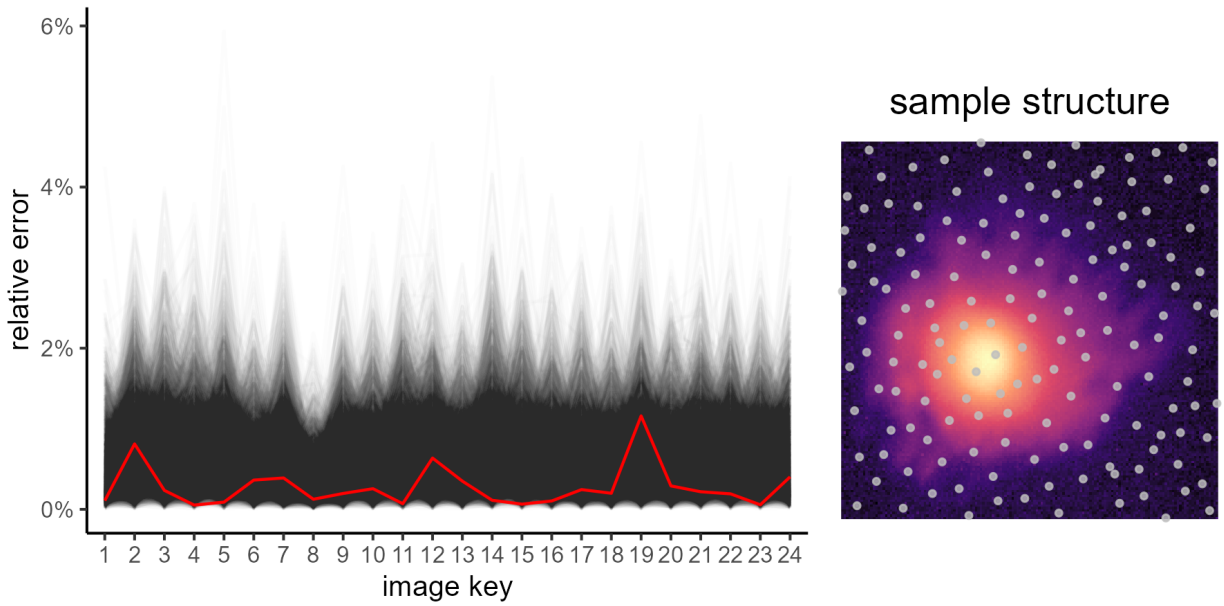

Figure 4.16

An illustration depicting the selection of the *optimal* sample structure and the selected structure.

selected sample structure. Each line in the left figure indicates one of the 10,000 sample structures evaluated. The red line indicates the sample structure that produced the lowest mean relative error.

4.4.2 Evaluating the Cost and Accuracy of the Update Procedure

Using tw_A, layers 2 through 16 were again used to form the training tensor, and CPD was applied with a rank four decomposition. The accuracy of the decomposition was captured using the RMSE. For the next 60 layers, the update procedure was applied along with a complete decomposition. To elaborate, let $step = 0$ indicate the initial decomposition applied to layers 2 through 16. Then $step = 1$ will update the initial decomposition to layers 3 through 17 and perform a new decomposition on layers 3 through 17. At $step = 2$ an update will be applied to the previously updated decomposition using layers 4 through 18 and a new decomposition on

those same layers. This continues to layers 62 through 76. The accuracy and computation time of both the update and the decomposition were collected at each layer. The average RMSE per step was 27.636 °C (95% CI [27.283, 27.989]) and 26.902 °C (95% CI [26.599, 27.204]) for the new decomposition and the update procedure, respectively. The average computation time was 9.744 seconds (95% CI [8.481, 11.008]) and 1.887 (95% CI [1.739, 2.035]) for the new decomposition[6] and the update procedure, respectively.

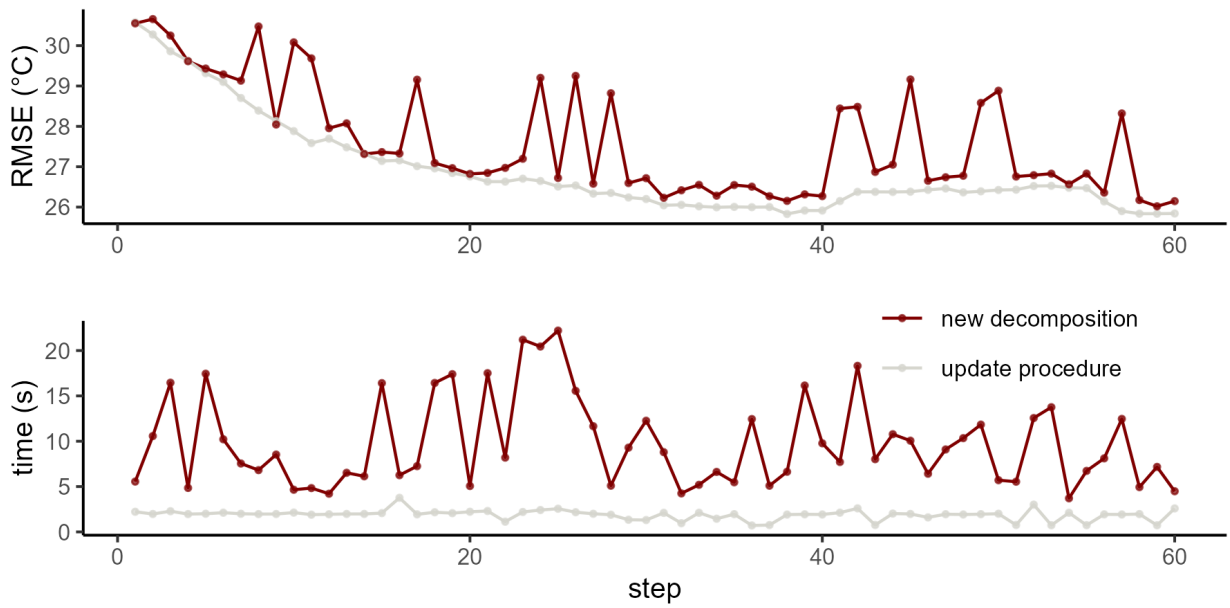

Figure 4.17

Performance comparison of the update procedure to a new decomposition.

Figure 4.17 illustrates a comparison between the update procedure to a new decomposition. The top image shows the RMSE per step while the bottom image shows the computation time per step.

[6]The disparity in the computation time reported here and that in table 4.8 was likely due to experiments in section 4.3 being run in parallel.

The update procedure consistently produces a more accurate decomposition and a significantly faster decomposition, with an average reduction in error of 2.583% (95% CI [1.812, 3.355]), and an average reduction in computation time of 76.160% (95% CI [72.708, 79.613]). Furthermore, a new decomposition was erratic in the amount of computation time required, with a standard deviation of 4.891 seconds, while the update procedure was more stable with a standard deviation of 0.573 seconds.

The majority of the computation time for the update procedure was for the inter-layer update which accounted for an average of 73.859% (95% CI [70.520, 77.198]) of the computation time. Figure 4.18 illustrates the breakdown of the computation time of the update procedure for the inter-layer and intra-layer updates.

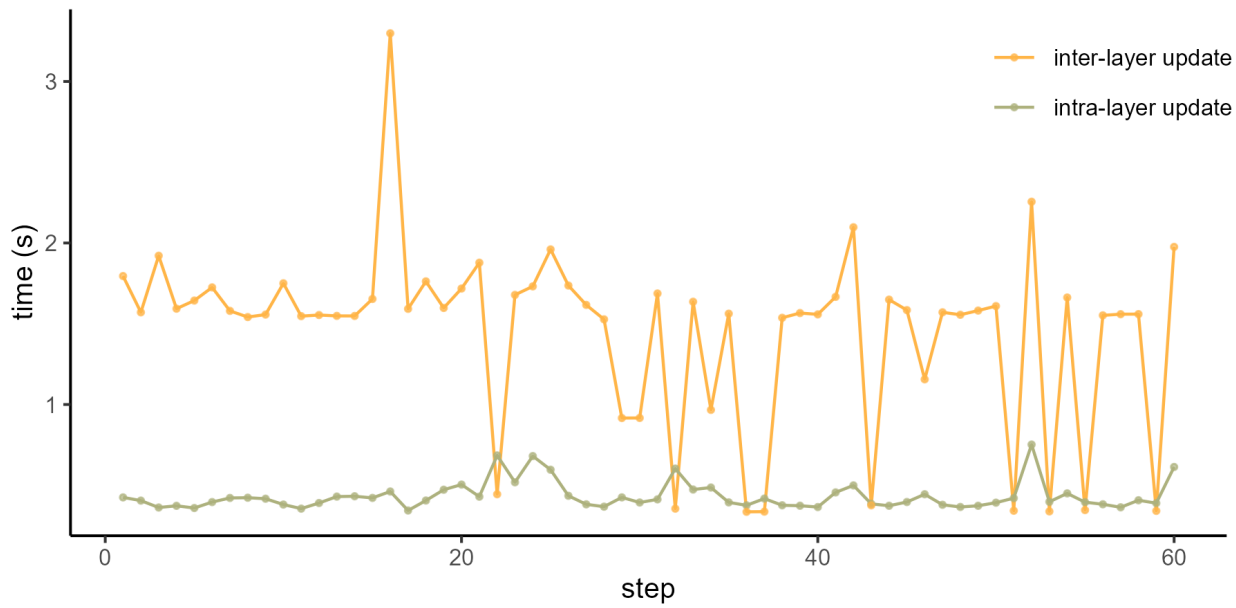

Figure 4.18

Comparing computation time of the intra-layer and inter-layer updates.

Instances where the inter-layer update requires less time than the intra-layer are those where the relative change in error were below the tolerance threshold.

4.4.3 Feasibility of Updating the NNAR Models

The purpose of this experiment was to investigate whether the lagged values of NNAR models could be modified using the updated tensor without much loss in accuracy. The test was performed in a similar manner as section 4.4.2 using the subsequent 30 layers for tensor updates.

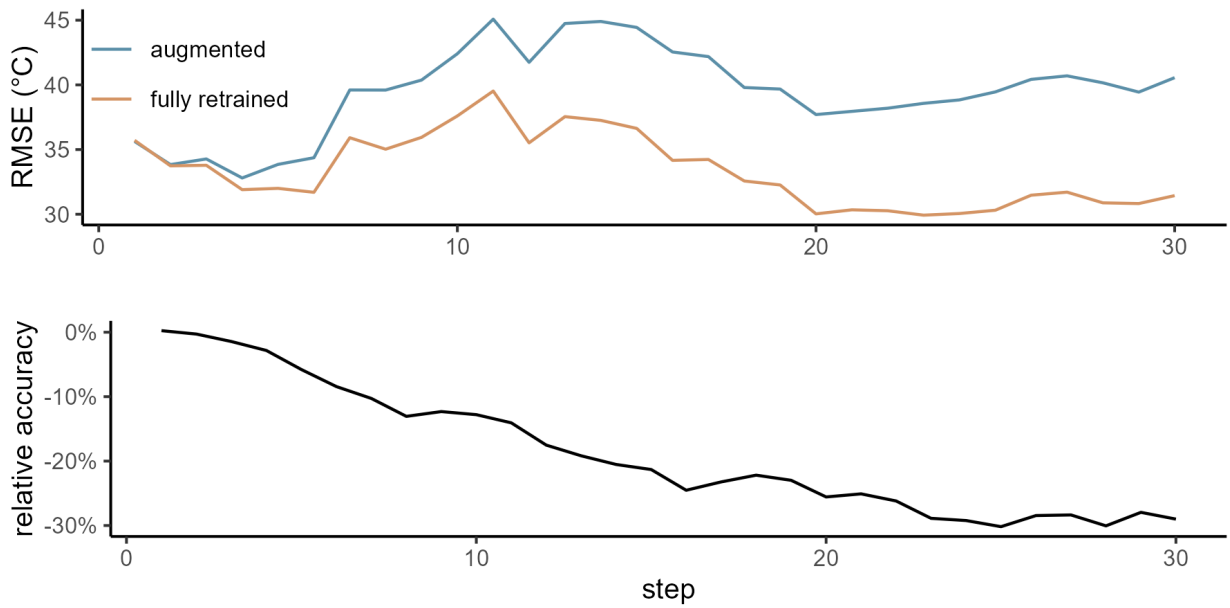

Figure 4.19

Comparing the accuracy of a fully retrained NNAR and an augmented NNAR.

A five layer forecast was calculated using a fully retrained NNAR model as done in section 4.3 and an augmented NNAR model. The augmented model was trained on the initial decomposition, then at each update, the lagged values were updated. The calculated RMSE for each of the two

models represents all five layers forecasted. Figure 4.19 provides an accuracy comparison between the two models. The top plot shows the five layer forecast RMSE for each model while the bottom plot shows the change in accuracy of the augmented model relative to the fully retrained model. Both models follow a similar pattern in their accuracy, but the augmented model consistently decreases in accuracy relative to the fully retained model for each layer updated. At 30 layers, the augmented model is nearly 30% less accurate than the fully retrained model. Regarding computation time, the augmented model is significantly faster, as expected. The fully retrained model had an average computation time of 9.405 seconds (95% CI [8.420, 10.391]) while the average computation time for the augmented model was 0.081 seconds (95% CI [0.072, 0.089]) resulting in a 99.142% reduction.

4.4.4 Discussion

The experiments in this section demonstrated that the tensor update methodology could significantly reduce the computation time to model the data with mixed results regarding accuracy. The update procedure reduced the decomposition cost by 76.2% and the model cost by 99.1%. The accuracy of the decomposition update was slightly improved over a new decomposition with error being reduced by 2.6%. However, the augmented NNAR had a steady decline in accuracy relative to a fully retrained NNAR. After 30 new layers were updated, the augmented NNAR had declined by 30%.

As the tensor decomposition accounts for nearly two-thirds of the computational cost, the 76.2% reduction while producing a more accurate estimate represents a success for the methodology. More investigation is needed to identify if the updated temporal components can be incorporated into a

previously trained NNAR without sacrificing much accuracy. As it currently stands, the reduction in accuracy of the augmented NNAR is likely not worth the savings in computational cost.

4.5 Transfer Learning Experiment

In this experiment, the decomposed tensor of the form used in model $T3_{4,15}$ of dataset tw_A is extended to tw_B using the transfer learning methodology outlined in section 3.4. Of interest in this experiment is the accuracy of the learned decomposition over time, the cost of the learning procedure, and feasibility of forecasting early in the transfer process.

4.5.1 Evaluation Procedure

A rank 4, 15 layer decomposition of layers 2-16 of tw_A are adjusted via the transfer learning methodology to newly observed data from tw_B. For brevity, this decomposed tensor will be refereed to as $\widehat{\mathcal{T}_A}$. As tw_A was formatted to have a consistent number of images per layer (24), the decomposed tensor is of dimensions $\widehat{\mathcal{T}_A} \in \mathbb{R}^{155\times155\times360}$.

Table 4.9

Number of images per layer for first 13 layers in tw_B.

Layer	1	2	3	4	5	6	7	8	9	10	11	12	13
$\mathbf{N}_i$	23	28	26	28	28	26	31	26	21	28	28	23	31

For this experiment, images were iteratively incorporated from tw_B using the first 360 images. Since tw_B has a varying number of images per layer and on average more images per layer, the first 360 images of tw_B include up to layer 14, with layer 14 incomplete. As updates are completed at

the end of each layer, the first 13 layers are evaluated which equates to the first 347 images. Table 4.9 lists the number of images $\mathbf{N}_i$ per layer for the first 13 layers for tw_B.

Using the notation adopted in section 3.4, let $\widehat{\mathcal{B}}_0 = \widehat{\mathcal{T}}_A[,,1{:}347]$ and $\mathcal{T}_B \in \mathbb{R}^{155\times155\times347}$ represent the tensor constructed from tw_B that is to be estimated and is used for evaluation in this experiment. Metrics on accuracy and computation time are captured after the addition of each layer.

4.5.2 Transfer Procedure Cost and Accuracy

Figure 4.20 illustrates the accuracy of the transfer as new layers are added.

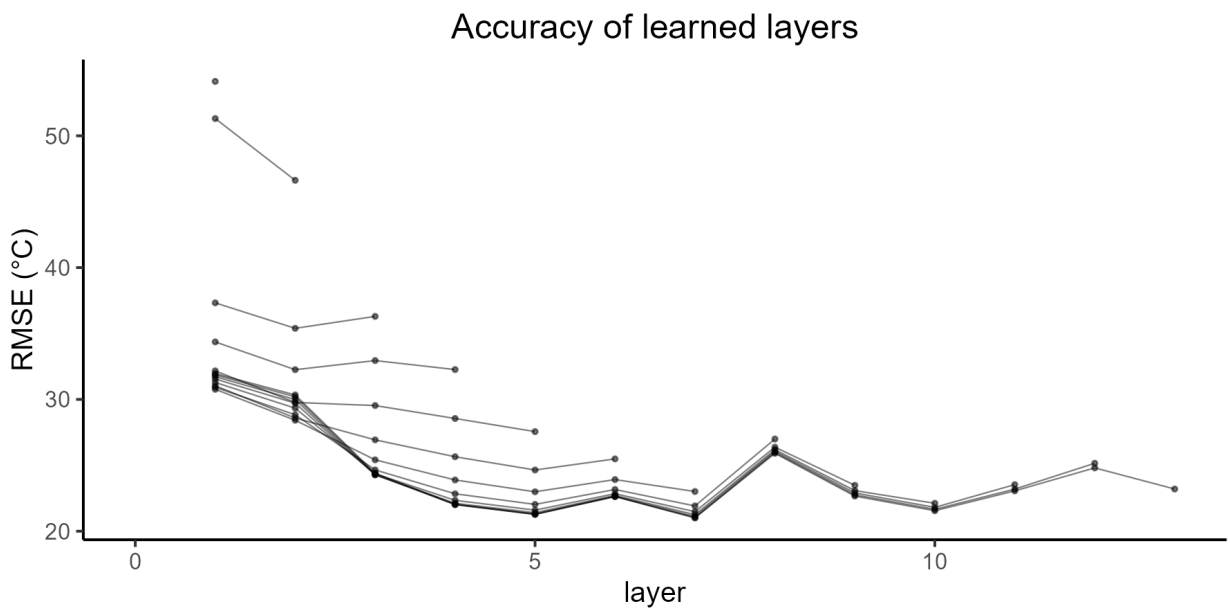

Figure 4.20

Accuracy of the transfer as layers are added.

Each line segment of length n depicts the accuracy at each layer after n layers. That is, the line segment of length 1 (single point) is the per layer RMSE of $\widehat{\mathcal{B}}_{23}[,,1:23]$ and $\mathcal{T}_B[,,1:23]$ while

the line segment of length 2 is the per layer RMSE of $\widehat{\mathcal{B}}_{51}[,,1:51]$ and $\mathcal{T}_B[,,1:51]$, etc, where the values 1:23 and 1:51 are the cumulative $\mathbf{N}_i$'s corresponding to the layer index. The first 2 layers of $\mathcal{T}_B$ contain unusual spatial characteristics indicated by the higher error compared to the other 11 layers. A significant decline in error occurs after the first 2 layers. By layer 3, small improvement in the approximation occurs with only marginal improvement occurring after layer 7. Figure 4.21 provides a specific example of the accuracy of the decomposition as layers are added for the first image in layer = 1. Residuals here are calculated as *predicted - actual*. Thus positive residuals indicate over prediction and negative residuals indicate under prediction.

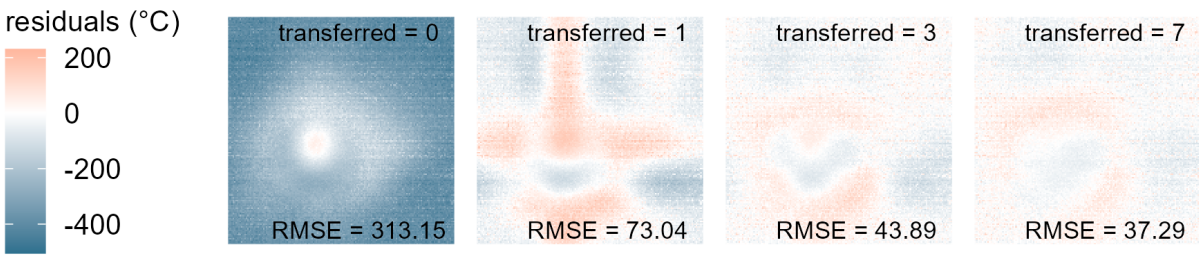

Figure 4.21

Accuracy of the transfer as layers are added for $\mathcal{T}_B[:,:,1]$.

Figure 4.22 illustrates the cost of the transfer update per layer. The second update requires the most time by a large margin at 15.036 seconds while all other updates require less than 3.7 seconds. The mean update time is 3.558 seconds (95% CI [1.452, 5.664]). Excluding the second update, the mean time is 2.601 seconds (95% CI [2.269, 2.933]).

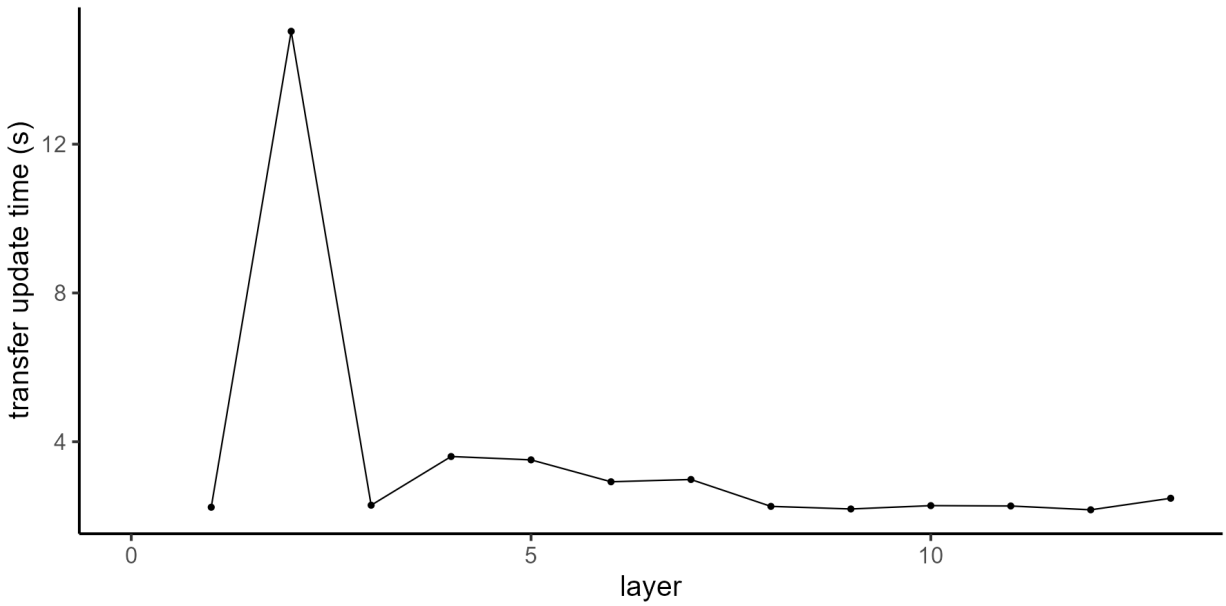

Figure 4.22

Computational cost of the transfer per layer.

To evaluate the feasibility of forecasting early in the transfer process, forecasts using an NNAR were examined. Since the NNAR requires at least 2 layers to incorporate layer effect, forecasts began at layer 3. Figure 4.23 illustrates the accuracy of the forecasts compared to the accuracy of mean-centering the previously learned temporal modes.

The left graph depicts the cumulative accuracy of the transfer with forecasts. Layer = 0 represents the RMSE calculated from the residuals of $\widehat{\mathcal{B}}_0$ and $\mathcal{T}_B$ before any modes were mean-centered. Layer = 1 represents the RMSE calculated from the residuals of $\widehat{\mathcal{B}}_{23}$ and $\mathcal{T}_B$ with the initial mean-centering. Layer = 2 to layer = 12 represents the RMSE calculated from the residuals of $\widehat{\mathcal{B}}_{51}$ and $\mathcal{T}_B$, $\widehat{\mathcal{B}}_{77}$ and $\mathcal{T}_B$, etc, with a comparison of NNAR forecasts and mean centering where forecasts begin at layer 3. Layer = 13 represents the RMSE calculated from the residuals of $\widehat{\mathcal{B}}_{347}$

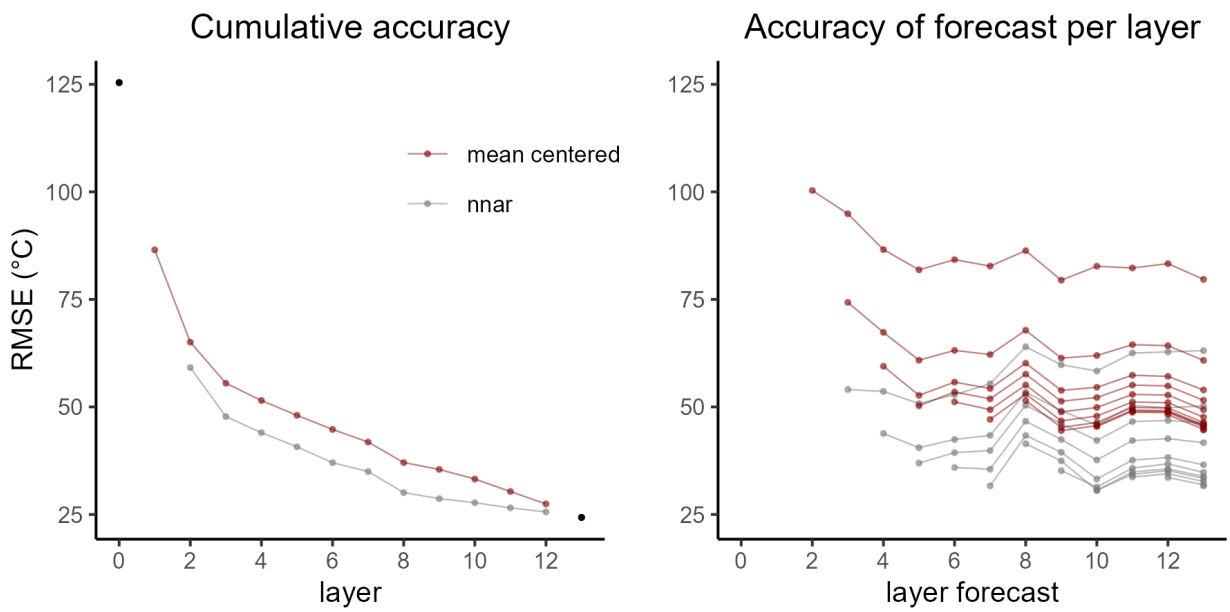

Figure 4.23

Illustration of accuracy of forecasts during transfer using NNAR and mean centered modes.

and $\mathcal{T}_B$ which represents the final learned decomposition. The right graph gives a similar visual as given in Figure 4.20. Here, the line segment represents the forecast accuracy at each layer. The line segment beginning at layer = n represents the forecasts made after learning the first $n - 1$ layers. It can be seen in both figures that the NNAR consistently has better performance than simply mean-centering. Additionally, the accuracy of the NNAR significantly increases up to layer 6. Forecasts beginning at layer 7 and beyond only have marginal improvements. The median RMSE per layer for an NNAR forecast from layer 2 is 58.347 °C, by layer 7 it is 35.238 °C, and by layer 10 it is 33.705 °C. Figure 4.24 provides a specific example of the accuracy of the NNAR forecast as layers are added using the last image in layer = 13. The NNAR models required an average of

2.229 seconds (95 % CI [1.616, 2.843]) to train and forecast the temporal modes. Here, again, residuals here are calculated as *predicted* - *actual*.

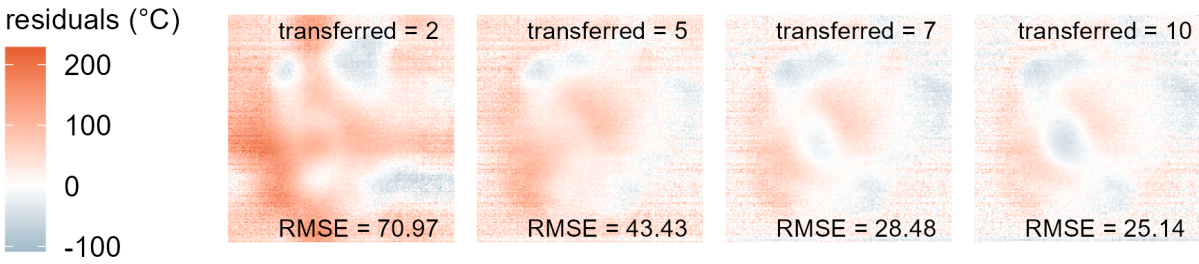

Figure 4.24

Accuracy of the forecast during transfer as layers are added for $\mathcal{T}_B[:, :, 347]$.

4.5.3 Discussion

The experiment in this section demonstrated that a previously learned decomposition can be quickly adjusted to data from a new process and that it is possible to make accurate forecasts early in the transfer process. By three layers, dominant spatial patterns within the data are captured and by seven layers, the decomposition only improves marginally meaning spatial patterns in the data are mostly learned by seven layers.

Regarding the forecast, an NNAR can begin making forecasts after two layers are learned. However, the accuracy of the forecasts increases significantly after six layers are learned, with only marginal improvements beyond that point meaning temporal patterns in the data are mostly learned by six layers.

CHAPTER V

CONCLUSION AND FUTURE WORK

One of the challenges that hinders more widespread adoption of AM technologies is the reliability of the fabricated parts. This reliability is dependent on the microstructural properties of a part which can be at least partially explained by the part's thermal history during its manufacture. The focus of this study was on a data-driven online modeling framework that could quickly approximate melt pool thermal history data, capturing the significant spatial and temporal influences, then produce accurate forecasts of the melt pool thermal history. The modeling framework was evaluated using two experimental datasets of melt pool thermal history, tw_A and tw_B, collected during the manufacture of two different thin walls via direct laser deposition. Some core challenges in modeling this data came from the noise present within the data and the dimensionality of the data. These problems were addressed by using tensor decomposition, namely canonical polyadic decomposition. This approach enabled identification of a low-rank structure of the data which both reduced noise and simplified the learning process for forecasting models by significantly reducing the size of the data.

The first experiment in this study evaluated a 4D representation of tw_A where images were stacked across track-position and layer. This formulation allowed layer-wise trends to be modeled explicitly using ARIMA models. Through a TSCV procedure, a rank 5 decomposition using 25

layers was identified as a good structure. The decomposition achieved a high-level of accuracy, capturing 97.54% of the variance in the data, while requiring an average of 24.651 seconds to learn the low-rank structure. Additionally, the forecasting models had good accuracy with an average RMSE of 29.701 °C and an NRMSE of 3.554% per image for 1-5 layers forecasted with the models requiring an average of 0.214 seconds to train. When compared to a seasonal naïve model, which had an RMSE and NRMSE of 38.454 °C and 4.601% respectively, the tensor model had a statistically significant increase in accuracy. While the 4D model was successful in accurately approximating and forecasting the thermal history, its rigid structure would pose an issue were it to be used to model thermal data from a part with a more complex geometry.

For the second experiment in this study, a 3D representation of tw_A was adopted to overcome the potential shortcoming of the 4D representation. In the 3D representation, images were stacked across time with the interaction of track-position and layer being captured by NNAR models. Through a TSCV procedure, a rank 4 decomposition using 15 layers was identified as a good structure. The 3D decomposition achieved a slightly higher level of accuracy, explaining 97.72% of the variance in the decomposition but had a slightly lower reduction in data size at 99.97%. The decomposition was faster requiring an average of 20.657 seconds due to the lower rank and smaller size. The forecasting models slightly outperformed in accuracy with an average RMSE and NRMSE per image of 29.323 °C and 3.481% respectively for 1-5 layers. However, the more complex NNAR required significantly more time to train averaging 10.670 seconds. Overall, the 3D representation succeeded in giving comparable results to the 4D representation while overcoming the 4D's rigid data structure requirement.

In the third experiment, an update procedure to the 3D decomposition was evaluated on tw_A. Given the initial decomposition, an in-situ process was simulated where as images were observed the learned parameters in the decomposition would be adjusted to fit the new data. This was performed in two procedures: an intra-layer update and an inter-layer update. The intra-layer update would be performed within each layer and would only adjust the temporal mode of the decomposition using a maximin LHS sample of the newly observed image while keeping the spatial modes fixed. The inter-layer update would be performed at the completion of each layer and would adjust all modes using the CP-algorithm where the modes are initialized with the previously learned values. The update procedure produced more accurate approximations than a new decomposition with an average reduction in error of 2.583%. Furthermore, the update procedure achieved a 76.160% reduction in the computation time of the decomposition with an average update time of 1.887 seconds per layer. In addition to the decomposition accuracy and computation time, the feasibility of augmenting an NNAR using the updated temporal mode opposed to fully retraining the NNAR was evaluated. While the augmented NNAR was significantly faster requiring an average of 0.081 seconds compared to an average of 9.405 seconds for the fully retrained NNAR, the accuracy declined at a consistent rate. After 30 layers, the augmented NNAR was approximately 30% less accurate than a fully retrained NNAR.

In the fourth experiment, a transfer learning procedure to the 3D decomposition was evaluated on tw_B. Given an initial decomposition trained on tw_A, an in-situ process was simulated where as images were observed from tw_B the learned parameters in the decomposition would be adjusted to fit the new data. This process used an augmented form of the update procedure to accomplish this process. The results of the experiment indicated that the transfer learning procedure could quickly

adjust to the new process by 3 layers, and by 7 layers the process is seemingly learned as only marginal improvement occurs thereafter. The computation time of the transfer was minor with one notable spike at layer 2 requiring 15.036 seconds. The average of the remaining twelve layers tested was 2.601 seconds. Forecasts during the transfer process using an NNAR could feasibly be made after 2 layers had been transferred. Error in the forecasts declined significantly after learning a few layers. The median RMSE per layer after transferring 2, 7, and 10 layers was 58.347 °C, 35.238 °C, and 33.705 °C respectively. The NNAR models required an average of 2.229 seconds to train and forecast during the transfer process.

The results of this research have demonstrated that the proposed modeling framework can accurately capture and forecast melt pool thermal history data in an online manner, and transfer a learned model to data from a new process. However, several opportunities exist for future work. Firstly, only two datasets were examined in this research with both datasets coming from thin wall builds. The modeling framework should be tested on builds using more complex geometries where the heat source will traverse in different directions. This will require additional preprocessing of the data, rotating the melt pool images to align in a single direction. An image registration technique like that used by Esfahani et al. [20] could be employed in this process. Additionally, it would be valuable to test the methodology on datasets collected from different metal printing technologies where monitoring the melt pool is feasible.

Secondly, limited attention was given to fine tuning the NNAR models in this study. Future research could explore hyperparameter tuning of the NNAR as well as other modeling strategies. Furthermore, no treatment was given to handling the missing images in tw_B and the model did not incorporate the time gap between the end of one layer and the beginning of another, i.e. the

temporal mode index was treated as the time step. Accounting for these discrepancies in the temporal model formulation should lead to greater accuracy. Additionally, feature engineering to identify more relevant variables for the NNAR could prove useful. For instance, a binary variable identifying the start of a new layer as a spike in temperature typically occurred at the start of a new layer. Lastly, anomaly detection based on decomposition accuracy and temporal patterns should be explored in greater detail. Appendix A offers an introduction to this approach demonstrating its viability, but a full treatment using multiple datasets from different builds should be explored.

REFERENCES

[1] D.-G. Ahn, "Directed energy deposition (DED) process: State of the art," *International Journal of Precision Engineering and Manufacturing-Green Technology*, vol. 8, 2021, pp. 703–742.

[2] O. Aljarrah, J. Li, A. Heryudono, W. Huang, and J. Bi, "Predicting part distortion field in additive manufacturing: a data-driven framework," *Journal of Intelligent Manufacturing*, vol. 34, no. 4, 2023, pp. 1975–1993.

[3] S. C. Altıparmak and B. Xiao, "A market assessment of additive manufacturing potential for the aerospace industry," *Journal of Manufacturing Processes*, vol. 68, 2021, pp. 728–738.

[4] Asharkyu, "A depiction of material extrusion,", https://d12oja0ew7x0i8.cloudfront.net/image-handler/ts/20220105085819/ri/750/src/images/news/ImageForNews_57817_16413910976555346.jpg, Unknown publication date, [Online; accessed August 10, 2023].

[5] ASTM Committee F42 on Additive Manufacturing Technologies. Subcommittee F42. 91 on Terminology, *Standard terminology for additive manufacturing technologies*, ASTM International, 2012.

[6] M. M. Bappy, C. Liu, L. Bian, and W. Tian, "In-situ Layer-wise Certification for Direct Energy Deposition Processes based on Morphological Dynamics Analysis," *Journal of Manufacturing Science and Engineering*, 2022, pp. 1–35.

[7] R. H. Bartels, J. C. Beatty, and B. Barsky, "An Introduction to the use of splines in computer graphics.," 1987.

[8] J. B. Bento, C. Wang, J. Ding, and S. Williams, "Process Control Methods in Cold Wire Gas Metal Arc Additive Manufacturing," *Metals*, vol. 13, no. 8, 2023, p. 1334.

[9] B. Berman, "3-D printing: The new industrial revolution," *Business horizons*, vol. 55, no. 2, 2012, pp. 155–162.

[10] C. M. Bishop et al., *Neural networks for pattern recognition*, Oxford university press, 1995.

[11] Y. Cai, J. Xiong, H. Chen, and G. Zhang, "A review of in-situ monitoring and process control system in metal-based laser additive manufacturing," *Journal of Manufacturing Systems*, vol. 70, 2023, pp. 309–326.

[12] D. D. Camacho, P. Clayton, W. J. O'Brien, C. Seepersad, M. Juenger, R. Ferron, and S. Salamone, "Applications of additive manufacturing in the construction industry–A forward-looking review," *Automation in construction*, vol. 89, 2018, pp. 110–119.

[13] R. Carnell, *lhs: Latin Hypercube Samples*, 2022, R package version 1.1.6.

[14] L. Chechik, A. D. Goodall, K. A. Christofidou, and I. Todd, "Controlling grain structure in metallic additive manufacturing using a versatile, inexpensive process control system," *Scientific Reports*, vol. 13, no. 1, 2023, p. 10003.

[15] F. Chen, M. Yang, and W. Yan, "Data-driven prognostic model for temperature field in additive manufacturing based on the high-fidelity thermal-fluid flow simulation," *Computer Methods in Applied Mechanics and Engineering*, vol. 392, 2022, p. 114652.

[16] A. Chergui, F. Villeneuve, N. Béraud, and F. Vignat, "Thermal simulation of wire arc additive manufacturing: a new material deposition and heat input modelling," *International Journal on Interactive Design and Manufacturing (IJIDeM)*, 2022, pp. 1–11.

[17] A. G. Dharmawan, Y. Xiong, S. Foong, and G. S. Soh, "A model-based reinforcement learning and correction framework for process control of robotic wire arc additive manufacturing," *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 4030–4036.

[18] L. Dowling, J. Kennedy, S. O'Shaughnessy, and D. Trimble, "A review of critical repeatability and reproducibility issues in powder bed fusion," *Materials & Design*, vol. 186, 2020, p. 108346.

[19] B. Duman and K. Özsoy, "A deep learning-based approach for defect detection in powder bed fusion additive manufacturing using transfer learning," *Journal of the Faculty of Engineering and Architecture of Gazi University*, vol. 37, no. 1, 2022, pp. 361–375.

[20] M. N. Esfahani, M. M. Bappy, L. Bian, and W. Tian, "In-situ layer-wise certification for direct laser deposition processes based on thermal image series analysis," *Journal of Manufacturing Processes*, vol. 75, 2022, pp. 895–902.

[21] S. Essongue, Y. Ledoux, and A. Ballu, "Speeding up mesoscale thermal simulations of powder bed additive manufacturing thanks to the forward Euler time-integration scheme: A critical assessment," *Finite Elements in Analysis and Design*, vol. 211, 2022, p. 103825.

[22] ExOne, "A depiction of binder jetting,", https://www.exone.com/ExOne/media/Case-Studies/ExOne-What-is-Binder-Jetting.jpg, Unknown publication date, [Online; accessed August 10, 2023].

[23] L. Fang, L. Cheng, J. A. Glerum, J. Bennett, J. Cao, and G. J. Wagner, "Data-driven analysis of process, structure, and properties of additively manufactured Inconel 718 thin walls," *npj Computational Materials*, vol. 8, no. 1, 2022, pp. 1–15.

[24] F. G. Fischer, M. G. Zimmermann, N. Praetzsch, and C. Knaak, "Monitoring of the powder bed quality in metal additive manufacturing using deep transfer learning," *Materials & Design*, vol. 222, 2022, p. 111029.

[25] P. Foteinopoulos, A. Papacharalampopoulos, and P. Stavropoulos, "On thermal modeling of Additive Manufacturing processes," *CIRP Journal of Manufacturing Science and Technology*, vol. 20, 2018, pp. 66–83.

[26] R. K. Ganeriwala, N. E. Hodge, and J. M. Solberg, "Towards improved speed and accuracy of laser powder bed fusion simulations via multiscale spatial representations," *Computational Materials Science*, vol. 187, 2021, p. 110112.

[27] J. Ge, J. Lin, Y. Lei, and H. Fu, "Location-related thermal history, microstructure, and mechanical properties of arc additively manufactured 2Cr13 steel using cold metal transfer welding," *Materials Science and Engineering: A*, vol. 715, 2018, pp. 144–153.

[28] I. Gibson, D. W. Rosen, B. Stucker, M. Khorasani, D. Rosen, B. Stucker, and M. Khorasani, *Additive manufacturing technologies*, vol. 17, Springer, 2021, pp. 1–9.

[29] I. Gibson, D. W. Rosen, B. Stucker, M. Khorasani, D. Rosen, B. Stucker, and M. Khorasani, *Additive manufacturing technologies*, vol. 17, Springer, 2021, pp. 469–470.

[30] I. Gibson, D. W. Rosen, B. Stucker, M. Khorasani, D. Rosen, B. Stucker, and M. Khorasani, *Additive manufacturing technologies*, vol. 17, Springer, 2021, pp. 164–165.

[31] I. Gibson, D. W. Rosen, B. Stucker, M. Khorasani, D. Rosen, B. Stucker, and M. Khorasani, *Additive manufacturing technologies*, vol. 17, Springer, 2021, p. 143.

[32] I. Gibson, D. W. Rosen, B. Stucker, M. Khorasani, D. Rosen, B. Stucker, and M. Khorasani, *Additive manufacturing technologies*, vol. 17, Springer, 2021, pp. 101–102.

[33] O. Gülcan, K. Günaydın, and A. Tamer, "The state of the art of material jetting—a critical review," *Polymers*, vol. 13, no. 16, 2021, p. 2829.

[34] S. Guo, W. Guo, and L. Bain, "Hierarchical spatial-temporal modeling and monitoring of melt pool evolution in laser-based additive manufacturing.," *IISE Transactions*, vol. 52, no. 9, 2020, pp. 977–997.

[35] S. Guo, C. Zamiela, and L. Bian, "Knowledge-transfer-enabled porosity prediction for new part geometry in laser metal deposition," *Journal of Manufacturing Processes*, vol. 103, 2023, pp. 64–77.

[36] G. He, T. Lu, H. Li, J. Lu, and H. Zhu, "Patch tensor decomposition and non-local means filter-based hybrid ASL image denoising," *Journal of Neuroscience Methods*, vol. 370, 2022, p. 109488.

[37] S. Ho, W. Zhang, W. Young, M. Buchholz, S. Al Jufout, K. Dajani, L. Bian, and M. Mozumdar, "DLAM: Deep Learning Based Real-Time Porosity Prediction for Additive Manufacturing Using Thermal Images of the Melt Pool," *IEEE Access*, vol. 9, 2021, pp. 115100–115114.

[38] F. Honarvar and A. Varvani-Farahani, "A review of ultrasonic testing applications in additive manufacturing: Defect evaluation, material characterization, and process control," *Ultrasonics*, vol. 108, 2020, p. 106227.

[39] X. Huang, T. Xie, Z. Wang, L. Chen, Q. Zhou, and Z. Hu, "A transfer learning-based multi-fidelity point-cloud neural network approach for melt pool modeling in additive manufacturing," *ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part B: Mechanical Engineering*, vol. 8, no. 1, 2022, p. 011104.

[40] R. J. Hyndman and G. Athanasopoulos, *Forecasting: principles and practice*, OTexts, 2018.

[41] R. J. Hyndman and Y. Khandakar, "Automatic time series forecasting: the forecast package for R," *Journal of Statistical Software*, vol. 26, no. 3, 2008, pp. 1–22.

[42] F. ILT, "A depiction of PBF,", https://www.3dnatives.com/en/wp-content/uploads/sites/2/DMLS1.jpg, Unknown publication date, [Online; accessed August 10, 2023].

[43] K. L. Johnson, T. M. Rodgers, O. D. Underwood, J. D. Madison, K. R. Ford, S. R. Whetten, D. J. Dagel, and J. E. Bishop, "Simulation and experimental comparison of the thermomechanical history and 3D microstructure evolution of 304L stainless steel tubes manufactured using LENS," *Computational Mechanics*, vol. 61, no. 5, 2018, pp. 559–574.

[44] M. Khanzadeh, S. Chowdhury, M. Marufuzzaman, M. A. Tschopp, and L. Bian, "Porosity prediction: Supervised-learning of thermal history for direct laser deposition," *Journal of manufacturing systems*, vol. 47, 2018, pp. 69–82.

[45] M. Khanzadeh, M. Dantin, W. Tian, M. W. Priddy, H. Doude, and L. Bian, "Fast prediction of thermal data stream for direct laser deposition processes using network-based tensor regression," *Journal of Manufacturing Science and Engineering*, vol. 144, no. 4, 2022.

[46] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM review*, vol. 51, no. 3, 2009, pp. 455–500.

[47] X. Kong, Y. Zhao, J. C.-W. Chan, and J. Xue, "Hyperspectral image restoration via spatial-spectral residual total variation regularized low-rank tensor decomposition," *Remote Sensing*, vol. 14, no. 3, 2022, p. 511.

[48] S. Koric and D. W. Abueidda, "Data-driven and physics-informed deep learning operators for solution of heat conduction equation with parametric heat source," *International Journal of Heat and Mass Transfer*, vol. 203, 2023, p. 123809.

[49] T. F. Lam, Y. Xiong, A. G. Dharmawan, S. Foong, and G. S. Soh, "Adaptive process control implementation of wire arc additive manufacturing for thin-walled components with overhang features," *The International Journal of Advanced Manufacturing Technology*, vol. 108, 2020, pp. 1061–1071.

[50] Y. Ledoux, S. Ghaoui, A. Ballu, C. Grandvallet, F. Villeneuve, M. Museau, F. Vignat, and T. H. Vo, "Fast simulation for powder bed fusion process based on thermal field pattern repetitions: application on electron beam melting process," *The International Journal of Advanced Manufacturing Technology*, 2023, pp. 1–10.

[51] J. Li, J. Bien, and M. T. Wells, "rTensor: An R Package for Multidimensional Array (Tensor) Unfolding, Multiplication, and Decomposition," *Journal of Statistical Software*, vol. 87, no. 10, 2018, pp. 1–31.

[52] W. Liang, J. Cao, L. Chen, Y. Wang, J. Wu, A. Beheshti, and J. Tang, "Crime Prediction With Missing Data Via Spatiotemporal Regularized Tensor Decomposition," *IEEE Transactions on Big Data*, 2023.

[53] T. Liu, M. Yuan, and H. Zhao, "Characterizing spatiotemporal transcriptome of the human brain via low-rank tensor decomposition," *Statistics in Biosciences*, vol. 14, no. 3, 2022, pp. 485–513.

[54] M. Markl and C. Körner, "Multiscale modeling of powder bed–based additive manufacturing," *Annu. Rev. Mater. Res*, vol. 46, no. 1, 2016, pp. 93–123.

[55] G. J. Marshall, S. M. Thompson, and N. Shamsaei, "Data indicating temperature response of Ti–6Al–4V thin-walled structure during its additive manufacture via Laser Engineered Net Shaping," *Data in brief*, vol. 7, 2016, pp. 697–703.

[56] R. Melentiev, N. Yu, and G. Lubineau, "Polymer metallization via cold spray additive manufacturing: A review of process control, coating qualities, and prospective applications," *Additive manufacturing*, vol. 48, 2021, p. 102459.

[57] S. Mohd Yusuf, S. Cutler, and N. Gao, "The impact of metal additive manufacturing on the aerospace industry," *Metals*, vol. 9, no. 12, 2019, p. 1286.

[58] M. Morup, D. M. Dunlavy, E. Acar, and T. G. Kolda, *Scalable tensor factorizations with missing data.*, Tech. Rep., Sandia National Laboratories (SNL), Albuquerque, NM, and Livermore, CA . . . , 2010.

[59] M. Mozaffar, S. Liao, H. Lin, K. Ehmann, and J. Cao, "Geometry-agnostic data-driven thermal modeling of additive manufacturing processes using graph neural networks," *Additive Manufacturing*, vol. 48, 2021, p. 102449.

[60] K. L. Ness, A. Paul, L. Sun, and Z. Zhang, "Towards a generic physics-based machine learning model for geometry invariant thermal history prediction in additive manufacturing," *Journal of Materials Processing Technology*, vol. 302, 2022, p. 117472.

[61] T. D. Ngo, A. Kashani, G. Imbalzano, K. T. Nguyen, and D. Hui, "Additive manufacturing (3D printing): A review of materials, methods, applications and challenges," *Composites Part B: Engineering*, vol. 143, 2018, pp. 172–196.

[62] I. Omle, A. H. Askar, E. Kovács, and B. Bolló, "Comparison of the Performance of New and Traditional Numerical Methods for Long-Term Simulations of Heat Transfer in Walls with Thermal Bridges," *Energies*, vol. 16, no. 12, 2023, p. 4604.

[63] H. Ouidadi, S. Guo, C. Zamiela, and L. Bian, "Real-time defect detection using online learning for laser metal deposition," *Journal of Manufacturing Processes*, vol. 99, 2023, pp. 898–910.

[64] S. K. Panda, S. Padhee, S. Anoop Kumar, S. S. Mahapatra, et al., "Optimization of fused deposition modelling (FDM) process parameters using bacterial foraging technique," *Intelligent information management*, vol. 1, no. 02, 2009, p. 89.

[65] V. Pandiyan, R. Drissi-Daoudi, S. Shevchik, G. Masinelli, T. Le-Quang, R. Loge, and K. Wasmer, "Deep transfer learning of additive manufacturing mechanisms across materials in metal-based laser powder bed fusion process," *Journal of Materials Processing Technology*, vol. 303, 2022, p. 117531.

[66] V. Petrovic, J. Vicente Haro Gonzalez, O. Jordá Ferrando, J. Delgado Gordillo, J. Ramón Blasco Puchades, and L. Portolés Griñan, "Additive layered manufacturing: sectors of industrial application shown through case studies," *International Journal of Production Research*, vol. 49, no. 4, 2011, pp. 1061–1079.

[67] T. Q. D. Pham, T. V. Hoang, X. Van Tran, Q. T. Pham, S. Fetni, L. Duchêne, H. S. Tran, and A.-M. Habraken, "Fast and accurate prediction of temperature evolutions in additive manufacturing process using deep learning," *Journal of Intelligent Manufacturing*, vol. 34, no. 4, 2023, pp. 1701–1719.

[68] S. D. Proell, P. Munch, W. A. Wall, and C. Meier, "A highly efficient computational framework for fast scan-resolved simulations of metal additive manufacturing processes on the scale of real parts," *arXiv preprint arXiv:2302.05164*, 2023.

[69] R Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2022.

[70] R Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2022.

[71] F. Radius, "A depiction of vat photopolymerization,", https://www.fastradius.com/wp-content/uploads/2021/01/Carbon-DLS-vat-photopolymerization-1024x576.jpg, January 4, 2021, [Online; accessed August 10, 2023].

[72] M. Reza Yavari, R. J. Williams, K. D. Cole, P. A. Hooper, and P. Rao, "Thermal modeling in metal additive manufacturing using graph theory: experimental validation with laser powder bed fusion using in situ infrared thermography data," *Journal of Manufacturing Science and Engineering*, vol. 142, no. 12, 2020, p. 121005.

[73] M. Roy and O. Wodo, "Data-driven modeling of thermal history in additive manufacturing," *Additive Manufacturing*, vol. 32, 2020, p. 101017.

[74] S. H. Seifi, W. Tian, H. Doude, M. A. Tschopp, and L. Bian, "Layer-wise modeling and anomaly detection for laser-based additive manufacturing," *Journal of Manufacturing Science and Engineering*, vol. 141, no. 8, 2019, p. 081013.

[75] N. Shamsaei, A. Yadollahi, L. Bian, and S. M. Thompson, "An overview of Direct Laser Deposition for additive manufacturing; Part II: Mechanical behavior, process parameter optimization and control," *Additive Manufacturing*, vol. 8, 2015, pp. 12–35.

[76] N. K. Shanthappa, R. H. Mulangi, and H. M. Manjunath, "The Spatiotemporal Patterns of Bus Passengers: Visualisation and Evaluation using Non-negative Tensor Decomposition," *Journal of Geovisualization and Spatial Analysis*, vol. 7, no. 1, 2023, p. 9.

[77] S. Shi, L. Wang, and X. Wang, "Uncovering the spatiotemporal motif patterns in urban mobility networks by non-negative tensor decomposition," *Physica A: Statistical Mechanics and its Applications*, vol. 606, 2022, p. 128142.

[78] S. Singh and S. Ramakrishna, "Biomedical applications of additive manufacturing: present and future," *Current opinion in biomedical engineering*, vol. 2, 2017, pp. 105–115.

[79] Stratasys, "A depiction of material jetting,", https://i.all3dp.com/workers/images/fit=scale-down,w=1456,gravity=0.5x0.5,format=auto/wp-content/uploads/2021/02/25112356/stratasys-J55-tech-illustration.jpg, Unknown publication date, [Online; accessed August 10, 2023].

[80] Y. Tang, M. R. Dehaghani, and G. G. Wang, "Review of transfer learning in modeling additive manufacturing processes," *Additive Manufacturing*, 2022, p. 103357.

[81] S. M. Thompson, L. Bian, N. Shamsaei, and A. Yadollahi, "An overview of Direct Laser Deposition for additive manufacturing; Part I: Transport phenomena, modeling and diagnostics," *Additive Manufacturing*, vol. 8, 2015, pp. 36–62.

[82] X. Tian, K. Xie, and H. Zhang, "A Low-Rank Tensor Decomposition Model With Factors Prior and Total Variation for Impulsive Noise Removal," *IEEE Transactions on Image Processing*, vol. 31, 2022, pp. 4776–4789.

[83] L. University, "A depiction of sheet lamination,", https://www.lboro.ac.uk/media/wwwlboroacuk/external/content/research/amrg/Sheet%20Lamination111.jpg, Unknown publication date, [Online; accessed August 10, 2023].

[84] C. V., "A depiction of DED,", https://www.3dnatives.com/en/wp-content/uploads/sites/2/DED_cover_1.jpg, September 10, 2019, [Online; accessed August 10, 2023].

[85] K. Vartanian, L. Brewer, K. Manley, and T. Cobbs, "Powder bed fusion vs. directed energy deposition benchmark study: Mid-size part with simple geometry," *Optomec, Tech. Rep*, vol. 1, 2016.

[86] C. Wang, X. Tan, S. Tor, and C. Lim, "Machine learning in additive manufacturing: State-of-the-art and perspectives," *Additive Manufacturing*, vol. 36, 2020, p. 101538.

[87] M. Wang, D. Hong, Z. Han, J. Li, J. Yao, L. Gao, B. Zhang, and J. Chanussot, "Tensor decompositions for hyperspectral data processing in remote sensing: A comprehensive review," *IEEE Geoscience and Remote Sensing Magazine*, 2023.

[88] H. Wei, T. Mukherjee, W. Zhang, J. Zuback, G. Knapp, A. De, and T. DebRoy, "Mechanistic models for additive manufacturing of metallic components," *Progress in Materials Science*, vol. 116, 2021, p. 100703.

[89] K. V. Wong and A. Hernandez, "A review of additive manufacturing," *International scholarly research notices*, vol. 2012, 2012.

[90] X. Xie, J. Bennett, S. Saha, Y. Lu, J. Cao, W. K. Liu, and Z. Gan, "Mechanistic data-driven prediction of as-built mechanical properties in metal additive manufacturing," *npj Computational Materials*, vol. 7, no. 1, 2021, pp. 1–12.

[91] Y. Yamaguchi and K. Hayashi, "Tensor Decomposition with Missing Indices.," *IJCAI*, 2017, vol. 17, pp. 3217–3223.

[92] Z. Yan, W. Liu, Z. Tang, X. Liu, N. Zhang, M. Li, and H. Zhang, "Review on thermal analysis in laser-based additive manufacturing," *Optics & Laser Technology*, vol. 106, 2018, pp. 427–441.

[93] C. Zamiela, Z. Jiang, R. Stokes, Z. Tian, A. Netchaev, C. Dickerson, W. Tian, and L. Bian, "Deep Multi-Modal U-Net Fusion Methodology of Thermal and Ultrasonic Images for Porosity Detection in Additive Manufacturing," *Journal of Manufacturing Science and Engineering*, vol. 145, no. 6, 2023, p. 061009.

[94] Y. Zhang, L. Wu, X. Guo, S. Kane, Y. Deng, Y.-G. Jung, J.-H. Lee, and J. Zhang, "Additive manufacturing of metallic materials: a review," *Journal of Materials Engineering and Performance*, vol. 27, 2018, pp. 1–13.

[95] M. Ziaee and N. B. Crane, "Binder jetting: A review of process, materials, and methods," *Additive Manufacturing*, vol. 28, 2019, pp. 781–801.

APPENDIX A

VIABILITY OF POROSITY DETECTION USING DECOMPOSITION

A.1 Introduction

In this chapter, a brief demonstration of anomaly detection using logistic regression without any hyperparameter tuning is given. While this analysis is limited, it serves as motivation for further development of detection schemes based on this tensor decomposition approach.

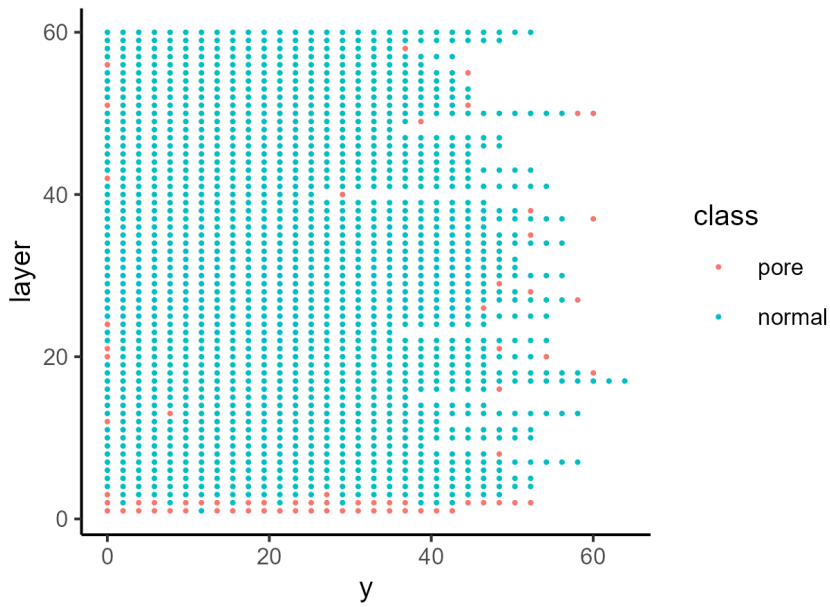

Figure A.1

Porosity locations in tw_B.

A.2 Data Description and Evaluation Procedure

tw_B, as outlined in section 4.1, was used in conjunction with X-ray CT scan data. The X-ray CT scan data was used to identify pores and their locations, with the location of each pore being assigned to the closest image in tw_B, and pore sizes ranging from 0.05 mm to 1.00 mm. Of the 1,564 images in tw_B, 1,486 images had no porosity, 71 images had porosity, and 7 images

were missing values. Figure A.1 illustrates the porosity locations in tw_B with the missing values removed. A class label of *pore* indicates porosity while *normal* indicates no porosity was detected. The reader can refer to Ho et al. [37] for a further discussion of the X-ray CT data.

Following the procedure employed in section 4.5, the decomposition learned from tw_A was applied to tw_B. The transfer learning procedure was used to fully update the learned decomposition using the first 360 images. The remaining images were incorporated using the update procedure without any drop off, that is, the final decomposition had a dimension of $155 \times 155 \times 1564$. The residuals per image were calculated then statistics on the residuals collected (RMSE, skewness, and kurtosis). These statistics, along with the learned temporal modes from the decomposition, were used as predictors for the logistic regression model. Five fold cross-validation with stratified sampling was used to evaluate the performance of the model.

Table A.1

Confusion matrix of cross-validation results.

		Ground Truth		
		pore	normal	Total
Prediction	pore	56	11	67
	normal	15	1,475	1,490
	Total	71	1,486	1,557

A.3 Results and Discussion

Table A.1 presents the confusion matrix for the cross-validation results. The logistic model had an accuracy of 98.33% ($\frac{1531}{1557}$) with a recall (true positive rate) of 78.87% ($\frac{56}{71}$) and a false positive

rate of 0.74% ($\frac{11}{1486}$). The area under the receiver operating characteristic (ROC) curve was 0.990. Figure A.2 illustrates the probability of a pore from the logistic model and the ROC curve.

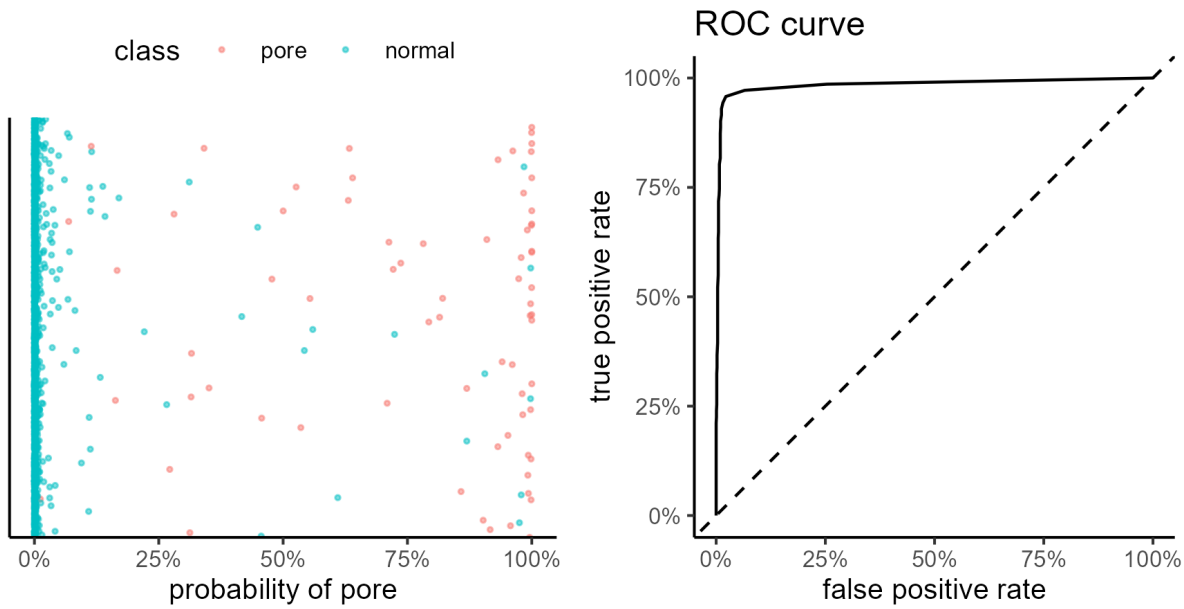

Figure A.2

Probability of pore and ROC curve.

In the left figure, the x-axis gives the predicted probability of a pore from the logistic model. A spread is added along the y-axis to aid in visualization. It can be seen in this figure that the majority of normal images have a probability close to 0 of containing a pore, whereas the pore images have probabilities spread from 0 to 1 with most near 1. In the right figure, the ROC curve illustrates the relationship between the false positive rate (FPR) and the true positive rate (TPR). As the classification threshold is moved from a higher probability to a lower probability, both the FPR and the TPR will increase. The steep incline of the ROC curve indicates that a significant

gain in the TPR can be had by lowering the classification threshold while having little impact on the FPR. Indeed, if the classification threshold is set to 28.07% for instance, the TPR is 98.99% and the FPR is 1.01%.

This brief demonstration provides strong evidence that an anomaly detection method driven by the tensor decomposition procedure is viable and should be further investigated. The strength of this approach comes from the ability to quickly gather the predictors, and an expected uniformity of most of these predictors as they are statistics of the residuals related to the decomposition. However, it will need to be investigated whether the learned temporal modes can serve as consistent predictors as they will vary given different decompositions, though a standardized version should offer the same explanatory power. Furthermore, a deeper investigation into the misclassifications should be performed as there is reason to believe the class labels could extend across layers. For instance, with the threshold set to 28.07% 15 locations are incorrectly identified as having pores (false positives). Of these 15 locations, 7 of the locations reside a layer above or a layer below the location classified as having a pore. If it is possible for a pore to be located across two layers, then it's effect on the thermal history may be seen at more than one location meaning the FPR may be overstated. Additionally, more complex models should be evaluated. A logistic regression model represents a high-bias classification model; models with greater flexibility should be explored and compared.

APPENDIX B

RELEVANT CODE

A listing of the relevant codes developed for this study. Note that %>% is the pipe operator from the MAGRITTR package. Unless otherwise noted using the double colon, function calls are from BASE R and DPLYR.

B.1 Model a Decomposed 4D Tensor

Description

Produce a model of a fourth-order decomposed tensor as described in 4.2.

Arguments

1. **U**: A list of the mode matrices calculated from the CP decomposition. Output of the CP function from the RTENSOR package.
2. **params**: A list of predictors associated with each mode.

Output

A list containing models for each mode of each component.

R Code

```
model4DTensor = function(U, params) {

        imgy_mods = apply(U[[1]], 2, FUN = function(y) {

                stats::splinefun(

                x = params[[1]], y = y, method = "natural"

                ) %>%

                return()

        })

        imgx_mods = apply(U[[2]], 2, FUN = function(y) {

                stats::splinefun(
```

```
        x = params[[2]], y = y, method = "natural"
        ) %>%
        return()
})
y_mods = apply(U[[3]], 2, FUN = function(y) {
        stats::splinefun(
        x = params[[3]], y = y, method = "natural"
        ) %>%
        return()
})
l_mods = apply(U[[4]], 2, FUN = function(y) {
        forecast::auto.arima(
        y, max.p = 10, max.q = 10,
        max.P = 10, max.Q = 10,
        max.order = 30, max.d = 5,
        stepwise = T, approximation = F
        ) %>%
        return()
})
modl = list(
imgy_mods = imgy_mods, imgx_mods = imgx_mods,
y_mods = y_mods, l_mods = l_mods
)
```

```
        return(modl)

}
```

B.2 Forecast Layers from 4D Model

Description

Forecast layers given model of a fourth-order decomposed tensor as described in 4.2.

Arguments

1. **tensor_mod**: A list containing models for each mode of each component. Output of the MODEL4DTENSOR function.
2. **n_layers**: The number of layers to forecast.
3. **lambdas**: The λ values obtained from the CP decomposition. Output of the CP function from the RTENSOR package.
4. **params**: A list of predictors for the forecast associated with each mode.

Output

A tensor providing the estimate for the next **n_layers**.

R Code

```
predictTensorLayers = function(tensor_mod, n_layers, lambdas, params) {

        A_hat = lapply(1:length(lambdas), function(i) {

                img_y = tensor_mod[[1]][[i]](params[[1]])

                img_x = tensor_mod[[2]][[i]](params[[2]])

                y_t   = tensor_mod[[3]][[i]](params[[3]])

                l = forecast::forecast(

                tensor_mod[[4]][[i]], h = n_layers

                )$mean %>%
```

```
                as.numeric()

                A_i = outer(img_y, img_x) %>%

                `*`(lambdas[i]) %>%

                outer(y_t) %>%

                outer(l) %>%

                rTensor::as.tensor()

        }) %>% Reduce(f = "+")

}
```

B.3 Model a Decomposed 3D Tensor

Description

Produce a model of a third-order decomposed tensor as described in 4.3.

Arguments

1. **U**: A list of the mode matrices calculated from the CP decomposition. Output of the CP function from the rTensor package.
2. **params**: A list of predictors associated with each mode.
3. **d**: The decay parameter for the neural network. A value of 10 was used in this study.
4. **maxit**: Maximum number of iterations to train the neural network.

Output

A list containing models for each mode of each component.

R Code

```
model3DTensor = function(U, params, d = 10, maxit = 100) {

        imgy_mods = apply(U[[1]], 2, FUN = function(y) {
```

```
            stats::splinefun(
            x = params[[1]], y = y, method = "natural"
            ) %>%
            return()
        })
        imgx_mods = apply(U[[2]], 2, FUN = function(y) {
            stats::splinefun(
            x = params[[2]], y = y, method = "natural"
            ) %>%
            return()
        })
        t_mods = apply(U[[3]], 2, FUN = function(y) {
            forecast::nnetar(
            y, xreg = params[[3]], repeats = 30,
            lambda = "auto", decay = d, maxit = maxit
            ) %>%
            return()
        })
        mod_list = list(
        imgy_mods = imgy_mods, imgx_mods = imgx_mods, t_mods = t_mods
        )
        return(mod_list)
}
```

B.4 Forecast Layers from 3D Model

Description

Forecast time steps ahead given model of a third-order decomposed tensor as described in 4.3.

Arguments

1. **tensor_mod**: A list containing models for each mode of each component. Output of the MODEL3DTENSOR function.
2. **n_steps**: The number of time steps to forecast.
3. **lambdas**: The λ values obtained from the CP decomposition. Output of the CP function from the RTENSOR package.
4. **params**: A list of predictors for the forecast associated with each mode.

Output

A tensor providing the estimate for the next **n_layers**.

R Code

```
predictTensor3D = function(tensor_mod, n_steps, lambdas, params) {

        A_hat = lapply(1:length(lambdas), function(i) {

                img_y = tensor_mod[[1]][[i]](params[[1]])

                img_x = tensor_mod[[2]][[i]](params[[2]])

                t_cor = forecast::forecast(

                tensor_mod[[3]][[i]], h = n_steps, xreg = params[[3]]

                )$mean %>%

                as.numeric()

                A_i = outer(img_y, img_x) %>%

                `*`(lambdas[i]) %>%
```

```
                outer(t_cor) %>%

                as.tensor()

        }) %>% Reduce(f = '+')

}
```

B.5 Loss Function for Intra-Layer Update

Description

Calculate the error given some temporal mode values and a sample structure.

Arguments

1. **par_vec**: A vector containing the estimated temporal mode values for one time step. The vector will be of length R, where R is the rank of the decomposition.
2. **old_tensor**: The tensor being updated.
3. **target_vec**: A vector containing the sampled values from the image to be estimated.
4. **tb_indices**: A data frame containing the indices of the sampled values.

Output

A scalar indicating the error.

R Code

```
tLossFunctionS = function(par_vec, old_tensor, target_vec, tb_indices) {

        lambdas = old_tensor$lambdas

        row_i = tb_indices$x1

        col_i = tb_indices$x2

        X_vec = lapply(1:length(lambdas), function(i) {

                img_y = old_tensor$U[[1]][,i][row_i]
```

```
            img_x = old_tensor$U[[2]][,i][col_i]

            t_cor = par_vec[i]

            img_y*img_x*lambdas[i]*t_cor

        }) %>% Reduce(f = '+')

        loss = norm(target_vec - X_vec, type = "2")

        return(loss)

}
```

B.6 Intra-Layer Update

Description

Estimate the temporal modes, with spatial modes fixed, given new data.

Arguments

1. **A_u**: The decomposed third-order tensor to be updated. Output of the CP function from the rTensor package.
2. **index_slice**: A vector of the indices to be updated. If a tensor update is being performed, indices will correspond to subsequent images beginning at 1. If tensor transfer, indices will correspond to learned mode.
3. **tb_sample_structure**: A data frame containing the indices of the sampled values.
4. **img_list**: A list containing the new data to be fitted.
5. **transfer**: Logical. Whether a tensor transfer is being performed or a tensor update.
6. **total_index**: A vector corresponding to the indices that have been updated, including **index_slice**. Used in the tensor transfer.

Output

A newly estimated temporal mode matrix.

R Code

```
temporalUpdate = function(
A_u, index_slice, tb_sample_structure,
img_list, transfer = F, total_index
) {
        U3 = A_u$U[[3]]
        R = ncol(U3); n = nrow(U3)
        for (j in index_slice) {
                if (transfer) {
                        if (j == 1) {
                                initial_vec = U3 %>%
                                as.data.frame() %>%
                                slice(j) %>%
                                unlist()
                        } else {
                                initial_vec = U3 %>%
                                as.data.frame() %>%
                                slice(j-1) %>%
                                unlist()
                        }
                } else {
                        initial_vec = U3 %>%
                        as.data.frame() %>%
```

```
            tail(1) %>%
            unlist()
        }
        target_vec = img_list[[j]] %>%
        reshape2::melt() %>%
        rename(x1 = Var1, x2 = Var2) %>%
        right_join(tb_sample_structure, by = c("x1", "x2")) %>%
        arrange(x1, x2) %>%
        pull(value)
        par_optim = optim(
        initial_vec, tLossFunctionS,
        old_tensor = A_u, target_vec = target_vec,
        tb_indices = tb_sample_structure
        )
        if (transfer) {
            U3[j, 1:R] = par_optim$par
        } else {
            U3 = rbind(
            U3[2:nrow(U3), 1:4],
            matrix(par_optim$par, nrow = 1)
            )
        }
}
```

```
        if (transfer) {
                for (k in 1:R) {
                        x_new = U3[total_index, k]
                        x_old = U3[(1:n)[-total_index], k]
                        mu_adj = mean(x_old) - mean(x_new, trim = 0.025)
                        x = (x_old - mu_adj)
                        U3[(1:n)[-total_index], k] = x
                }
        }
        return(U3)
}
```

B.7 Inter-Layer Update

Description

Reestimate the CP decomposition with initialized values. Code adapted from the CP function from the RTENSOR package [51].

Arguments

1. **A_cp**: The decomposed third-order tensor to be updated. Output of the CP function from the RTENSOR package.
2. **A_new**: A tensor containing the newly available data.
3. **max_iter**: The maximum number of iterations the ALS algorithm should be applied.
4. **tol**: The tolerance for the relative change in error between updates.

Output

A decomposed tensor.

R Code

```
updateCP = function (A_cp, A_new, max_iter = 25, tol = 1e-05) {
        num_modes    = A_cp$est@num_modes
        modes        = A_cp$est@modes
        U_list       = A_cp$U
        tnsr_norm    = fnorm(A_new)
        unfolded_mat = lapply(1:num_modes, function(m) rs_unfold(A_new, m = m)@data)
        Z = rTensor:::.superdiagonal_tensor(
        num_modes = num_modes, len = ncol(U_list[[1]]), elements = A_cp$lambdas
        )
        est = ttl(Z, U_list, ms = 1:num_modes)
        curr_iter   = 2
        converged   = FALSE
        fnorm_resid = rep(0, max_iter)
        fnorm_resid[1] = A_cp$fnorm_resid
        CHECK_CONV = function(est) {
                curr_resid = fnorm(est - A_new)
                fnorm_resid[curr_iter] <<- curr_resid
                if (curr_iter == 1) {
                        return(FALSE)
                }
                if (abs(curr_resid - fnorm_resid[curr_iter - 1])/tnsr_norm < tol) {
                        return(TRUE)
```

```
        } else {
                return(FALSE)
        }
}
pb = txtProgressBar(min = 0, max = max_iter, style = 3)
norm_vec = function(vec) norm(as.matrix(vec))
while ((curr_iter < max_iter) && (!converged)) {
        setTxtProgressBar(pb, curr_iter)
        if (curr_iter == 2) {
                if (CHECK_CONV(est)) {
                        converged <- TRUE
                        setTxtProgressBar(pb, max_iter)
                        lambdas = A_cp$lambdas
                } else {
                        curr_iter <- curr_iter + 1
                }
        } else {
                for (m in 1:num_modes) {
                        V <- hadamard_list(lapply(U_list[-m], function(x) {
                                t(x) %*% x
                        }))
                        V_inv <- solve(V)
                        tmp <- unfolded_mat[[m]] %*% khatri_rao_list(
```

```
                        U_list[-m],
                        reverse = TRUE
                        ) %*% V_inv
                        lambdas <- apply(tmp, 2, norm_vec)
                        U_list[[m]] <- sweep(tmp, 2, lambdas, "/")
                        Z = rTensor:::.superdiagonal_tensor(
                        num_modes = num_modes,
                        len = ncol(U_list[[1]]),
                        elements = lambdas
                        )
                        est <- ttl(Z, U_list, ms = 1:num_modes)
                }
                if (CHECK_CONV(est)) {
                        converged <- TRUE
                        setTxtProgressBar(pb, max_iter)
                } else {
                        curr_iter <- curr_iter + 1
                }
        }
        if (!converged) {
                setTxtProgressBar(pb, max_iter)
        }
}
```

```
        close(pb)
        fnorm_resid <- fnorm_resid[fnorm_resid != 0]
        norm_percent <- (1 - (tail(fnorm_resid, 1)/tnsr_norm)) * 100
        invisible(
        list(
        lambdas = lambdas, U = U_list, conv = converged,
        est = est, norm_percent = norm_percent,
        fnorm_resid = tail(fnorm_resid, 1), all_resids = fnorm_resid
        )
        )
}
```