wbk
Institut für Produktionstechnik

Master's Thesis

Emrecan Ilik
Matriculation no. 2316543

Process Monitoring in Additive Manufacturing through Machine Learning

wbk
Institute of Production Science
Karlsruhe Institute of Technology (KIT)
Kaiserstraße 12
76131 Karlsruhe

Prof. Dr.-Ing. Jürgen Fleischer
Prof. Dr.-Ing. Gisela Lanza
Prof. Dr.-Ing. habil. Volker Schulze

Statement of Originality

I sincerely affirm to have composed this thesis work autonomously, to have indicated completely and accurately all aids and sources used and to have marked anything taken from other works, with or without changes. Furthermore, I affirm to have observed the constitution of the KIT for the safeguarding of good scientific practice, as amended.

Karlsruhe, den 2023-11-15

Emrecan Ilik

Kurzfassung

Industrie 4.0 bezeichnet den Trend zur Nutzung von datenzentrierten und automatisierten Systemen in der Produktionstechnik, der durch Technologien wie das Internet der Dinge und künstliche Intelligenz ermöglicht wird. Mit der Verfügbarkeit von Daten und leistungsfähiger Hardware kann maschinelles Lernen komplexe Probleme in der Produktion bewältigen.

Darüber hinaus wird die additive Fertigung direkt bei der Herstellung von Produkten eingesetzt. Allerdings stellen Prozessinkonsistenzen, die aufgrund der Natur dieser Produktionstechnologie auftreten, ein Hindernis dar. Folglich ist es schwierig, die additive Fertigung zur Herstellung hochbelasteter, festigkeitskritischer Teile zu nutzen. Um diesem Problem zu begegnen, wird auf der Basis des Pulverbettschmelzens von Metallen mittels Laserstrahltechnik eine Technologie zur Prozessüberwachung entwickelt, bei der Poren in additiv gefertigten Bauteilen über akustische Signale während des Fertigungsprozesses detektiert werden. Ziel dieser Studie ist die Detektion von Poren im Bereich der additiven Fertigung durch die Verarbeitung akustischer Signale mit Hilfe von Machine-Learning Methoden.

Abstract

Industry 4.0 denotes the trend toward utilizing data-centric, automated systems in production technologies facilitated by cutting-edge technologies like the Internet of Things and artificial intelligence. With the availability of data and strong hardware, machine learning can handle complex problems in production.

Additionally, additive manufacturing is being directly used in the construction of products. However, process inconsistencies that occur due to this production technology's nature pose obstacles. Consequently, it is difficult to utilize additive manufacturing to construct highly stressed, strength-critical parts. To address this problem, a technology for process monitoring is being developed on the basis of powder bed fusion of metals using laser beam technology, whereby pores in additive-manufactured parts via acoustic signals during manufacturing processes are detected. This study aims to detect pores in the field of additive manufacturing by processing acoustic signals using machine-learning methods.

Contents

Abbreviations

AI	Artificial Intelligence
AM	Additive Manufacturing
CNN	Convolutional Neural Network
Conv1d	1-Dimensional Convolution
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
MaxPool1d	1-Dimensional Max Pooling
ML	Machine Learning
PCA	Principal Component Analysis
ReLU	Rectified Linear Unit
ResNet	Residual Neural Network
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent
SMOTE	Synthetic Minority Oversampling Technique
UMAP	Uniform Manifold Approximation and Projection for Dimension Reduction

1 Introduction

1.1 Motivation

The fourth industrial revolution denotes the current movement toward intelligent automation technologies. Problems in modern industry have been growing. Therefore, intelligent systems are needed to address complex problems. In the field of artificial intelligence (AI), deep learning provides good solutions (Hernavs et al., 2018).

Moreover, both in the field of research and industry, there is an agreement that additive manufacturing (AM) is an important technology that demonstrates high capability and a big deployment potential that conventional manufacturing techniques cannot match. Additionally, constructing high-quality and repeatable parts through AM is really a difficult task because of various factors (Tapia & Elwany, 2014).

On top of that, according to Yang et al. (2017) implementation of machine learning (ML) methods has showcased efficiency in numerous fields such as computer science, manufacturing, and health care. Additionally, Lasi et al. (2014) indicates that the advent of data acquisition and storage techniques facilitated the utilization of data-centric methods on top of ML technologies to discover hidden knowledge and complex relationships in digital manufacturing systems. Furthermore, Qin et al. (2022) points out that in the domain of process optimization and quality control ML models can learn hidden patterns and unearth latent knowledge to foster better decision-making.

Furthermore, to tackle process inconsistencies in additive manufacturing, a technology for process monitoring is being developed on the basis of powder-based fusion of metals using laser beam technology in order to predict possible defects via acoustic signals during manufacturing processes. To predict the defects, this study aims to use machine learning models including autoencoders, multi-layer perception, long short-term memory, and convolutional neural networks.

1.2 Objective

This study aims to predict pores in additive-manufactured parts through machine learning and deep learning by processing acoustic signals. To accomplish this, this study focuses on feature engineering methods in order to compress and transform raw signal data. Additionally, deep learning methodologies including autoencoders, convolutional neural networks, and long short-term memory will be utilized to identify and forecast the pores.

We will deal with two problem sorts, namely regression and classification. In regression, we will try to predict continuous labels. In classification, we aim to predict pores with high accuracy. We will evaluate the machine learning models based on evaluation metrics like mean squared error, accuracy, and F1-score.

Overall, literature research on machine learning and pore detection in additive manufacturing, evaluation and selection of machine learning models, customization of machine learning and data preprocessing models in accordance with the signal data, and evaluation of the training results constitute the objectives of this study.

1.3 Structure of the Thesis

This section discusses the organization of the study. Chapter 1 delves into the motivation behind this study, and the objective that it pursues. Chapter 2 highlights the context of additive manufacturing and machine learning methodologies. It explains the concept of additive manufacturing, machine learning, and deep learning algorithms that were used in this study.

Furthermore, Chapter 3 discusses the state of the art regarding porosity detection and data compression using autoencoders. In this section, the literature that revolves around porosity and deformity detection in additive manufacturing and in general production environments is discussed. Then, autoencoder models that are used to compress data and signal processing will be explored.

Chapter 4 focuses on the approach of this study. First, the dataset will be explored and its characteristics will be discussed. Then, the methodology regarding data preprocessing and machine learning pipeline will be introduced.

Chapter 5 sheds light on the training results of the machine learning models. The results will be discussed by providing visualizations of the training progress of the machine learning models.

Chapter 6 revolves around the evaluation of the results. In Chapter 6, conclusions regarding the training results will be discussed. Chapter 7 sums up the conclusions and outcomes of this study, and offers ideas to improve future work.

2 Foundations

2.1 Additive Manufacturing

Additive manufacturing (AM) has emerged since the 1980s as a solution to address the need for shorter production times, enhanced product quality, and product development flexibility. In additive manufacturing products are constructed layer-by-layer directly from 3D CAD models (Godec et al., 2022).

Furthermore, the study by Cooper (2001) states that rapid prototyping technologies are not solely being used for constructing models, rather it has been possible to produce final goods by using those manufacturing methodologies. Currently, these technologies have names like 3D printing, yet they stem from rapid prototyping (Noorani, 2006; Kochan, 1997). In Figure 2.1, stages of product development using prototyping technologies are represented.

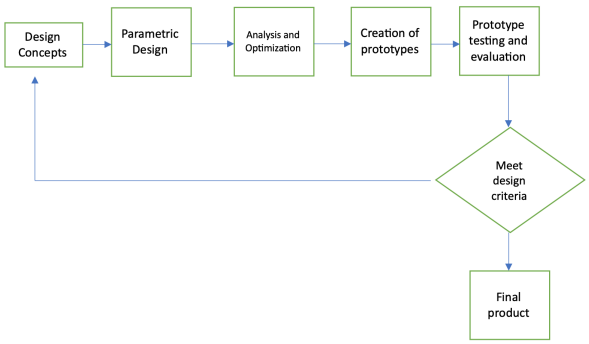

Figure 2.1: Product development steps - Noorani (2006)

Layer-by-layer construction of products using 3D CAD models enables quick modification regarding product design and low production volumes. Furthermore, additive manufacturing makes the construction of complex parts possible which can not be achieved by conventional production methods All in all, the additive manufacturing process is composed of three phases (Abdulhameed et al., 2019). Figure 2.2, depicts the phases of additive manufacturing.

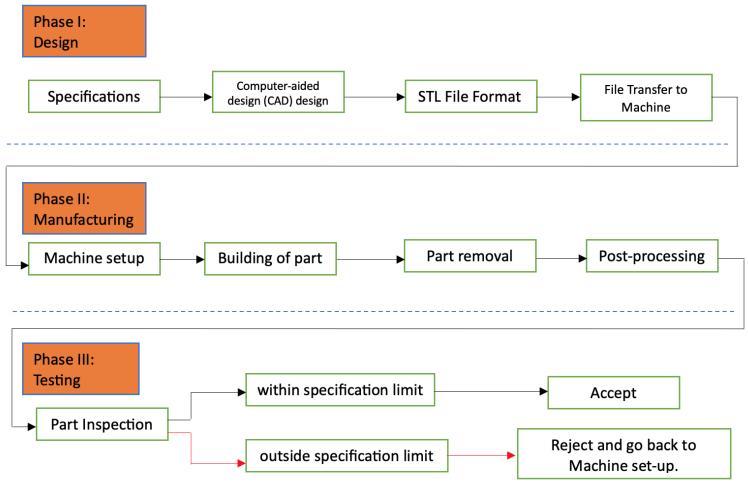

Figure 2.2: Additive Manufacturing Phases - Abdulhameed et al. (2019)

Currently, additive manufacturing is considered a rapid tooling, and a manufacturing method. Moreover, it is one of the significant elements of the fourth industrial revolution owing to its flexible nature that enables the customization of products. Additive manufacturing focuses on value-added products that can be quickly constructed which are limited in quantity (de Leon et al., 2016).

According to Prakash, Nancharaih & Rao (2018) additive manufacturing technology is composed of five steps:

1. Development of 3D CAD model
2. Transformation into a standard AM format (Kumar & Dutta, 1997; Huang et al., 2013)
3. An AM machine receives the file to process it
4. The AM machine constructs the part layer-by-layer
5. Completing the model

2.2 Machine Learning

In the age of big data, the importance of data in the context of business, and also for individuals has increased. Therefore, algorithms that can process the data are needed. Machine learning is a subfield in artificial intelligence that can deal with complex problems by generating algorithms based on data characteristics (Alpaydin, 2020).

Besides being implemented in various domains of industries, machine learning impacts empirical sciences like cosmology and biology as it can analyze high amounts of experimental data through cutting-edge methods. In machine learning, the learning paradigm concerns enhancing task performance via training experiences (Jordan & Mitchell, 2015). Figure 2.3 depicts life cycle of AI.

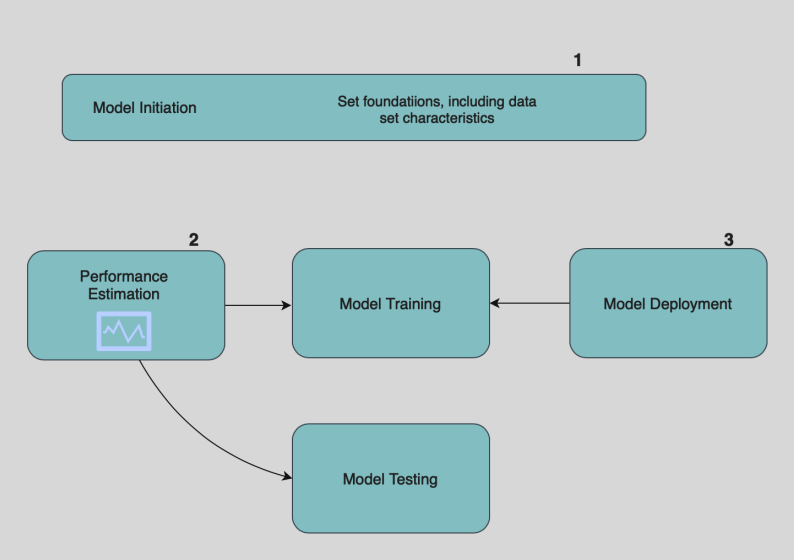

Figure 2.3: The AI Lifecycle - Kühl et al. (2021)

To put it briefly, one can define machine learning as a domain in computer science that strives to empower algorithms to learn autonomously without requiring explicit programming (Samuel, 1959).

Furthermore, machine learning (ML) is commonly categorized according to whether the learning process of the ML algorithm is supervised or unsupervised. In supervised learning, the value of each dependent variable is called its label, thus labeled data refers to data with specific values. Decision trees, support vector machines, and linear and logistic regression are among the supervised learning methods (Bi et al., 2019).

Moreover, unsupervised learning algorithms aim to identify patterns in the data without pursuing the objective of reaching a correct outcome. Unsupervised learning methods are similar to statistical techniques that strive to recognize unidentified classes that possess similar properties. For instance, clustering algorithms are categorized as unsupervised learning methods as they pursue the objective of grouping data instances based on a similarity metric. Other examples can be k-means clustering and Gaussian mixture methods (Duda, Hart et al., 1973; Bartholomew, Knott & Moustaki, 2011; Hennig et al., 2015; Bishop & Nasrabadi, 2006).

Additionally, it is worth mentioning that labeling data is a costly and time-consuming process. Semisupervised learning algorithms can train models using both labeled and unlabeled data (Zhu & Goldberg, 2022).

Beyond that reinforcement learning deals with associating situations with policies. Thus, a reinforcement learning agent pursues the objective of maximizing a reward signal. The agent is not taught which action to take, yet it learns which action yields the highest reward through trial and error (Sutton & Barto, 2018).

2.2.1 Deep Learning

Traditional machine learning methods encounter difficulties in processing raw data. On the other hand, deep learning can overcome this difficulty by using multiple layers to transform the data from its raw form into more abstract ideas (LeCun, Bengio & Hinton, 2015; Bai et al., 2021). Figure 2.4 depicts an example of a deep neural network architecture.

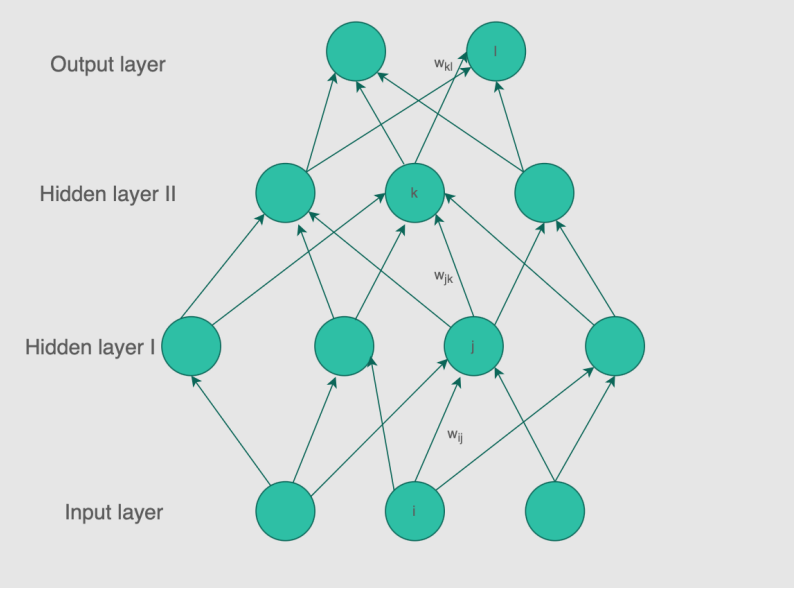

Figure 2.4: A Deep Neural Network - LeCun, Bengio & Hinton (2015)

Feature extraction is a significant part of obtaining meaningful information from large data sets. Traditional machine learning algorithms rely on well-defined features. Manual feature design is a time-consuming, and difficult process because it requires domain expertise and the implementation of specific methodologies. By contrast, deep neural networks cope with the constraints of artisanal future engineering through their sophisticated architecture which enables deep neural networks to systemize learning features without exerting a high level of labor (Goodfellow, Bengio & Courville, 2016; Janiesch, Zschech & Heinrich, 2021).

In Figure 2.5 the comparison of model-building procedures in explicit programming, shallow machine learning, and deep learning are depicted.

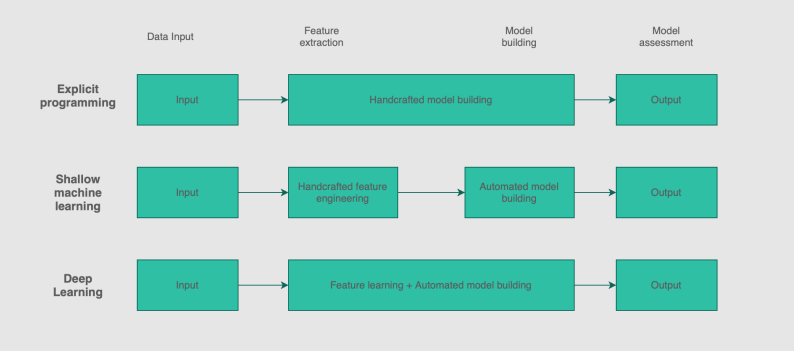

Figure 2.5: Comparison of Model Building Procedures - Goodfellow, Bengio & Courville (2016), Janiesch, Zschech & Heinrich (2021)

To conclude, deep learning differs from traditional machine learning in that it does not rely on manual feature extraction, but instead, it trains on large amounts of data. If enough quantity of input is provided, it can learn the features of the data without requiring a high level of domain knowledge in development (Huang et al., 2023). Figure 2.6 compares deep learning to traditional machine learning.

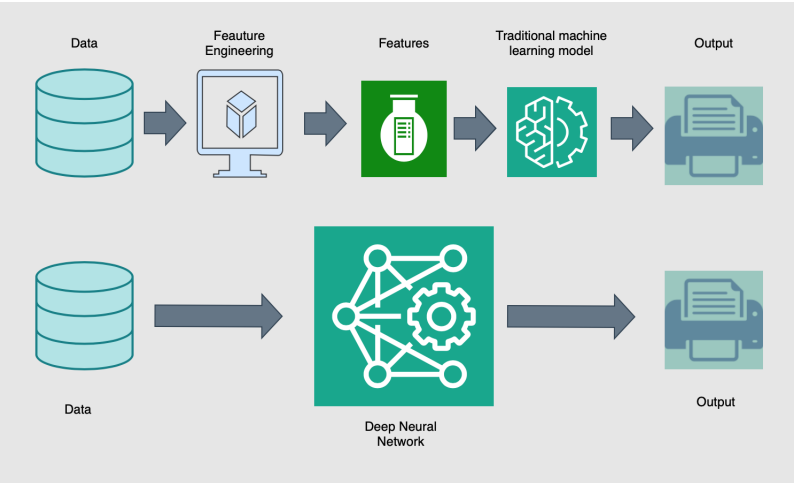

Figure 2.6: Deep Learning Compared to Traditional Machine Learning - Huang et al. (2023)

2.2.2 Autoencoders

Oftentimes, neural networks are used in supervised learning wherein each data instance x_i is associated with a label y_i. Consequently, the neural network model learns the association between training data and labels. Imagine that we have only a training dataset that encompasses M observations $S_T = x_i$ for $i = 1, 2, \ldots, M$. In the pursuit of reconstructing the training dataset with minimal error, autoencoders were first proposed by Rumelhart, Hinton & Williams (1986). Hence, a question arises: Why would it be beneficial to reconstruct training instances? The answer lies in the definition of autoencoder. An autoencoder is sort of an algorithm that pursues the objective of identifying descriptive information of the data through learning to reconstruct a part of input instances adequately (Michelucci, 2022; Rumelhart, Hinton & Williams, 1986; Bank, Koenigstein & Giryes, 2023).

An autoencoder is composed of three elements, namely an encoder model, a latent feature representation, and a decoder model. An autoencoder should be able to reconstruct the input data adequately. Simultaneously, it can also generate a latent representation that is meaningful. The latent representation can be beneficial for numerous tasks like feature extraction that can be utilized later for performing predictions (Michelucci, 2022). Figure 3.2 depicts the architecture of a standard autoencoder.

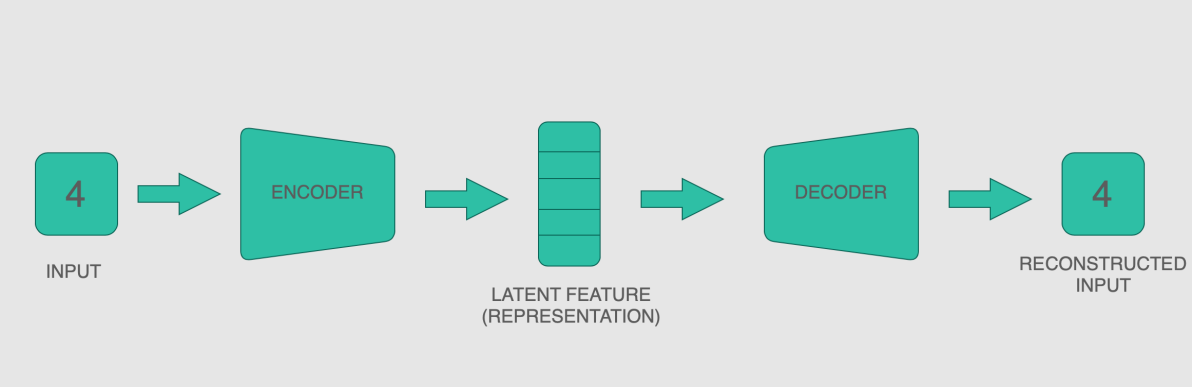

Figure 2.7: General Architecture of an Autoencoder - Michelucci (2022)

For the purpose of reducing the dimension of the input data, a bottleneck is enforced. The bottleneck maps the input data to a lower-dimension representation, which serves for data compression and feature extraction. The bias-variance trade-off is an important obstacle in training autoencoders. A proper autoencoder model should be able to reconstruct the input data well. In other words, it should represent a low reconstruction error. Furthermore, the compressed data should contain the general pattern in the original input. To tackle the trade-off issue, one prune option is to promote sparsity in hidden activations. Sparsity can be facilitated by incorporating it at the bottleneck layer, or as an alternative to it. One of the options to impose sparsity is to utilize L_1 regularization (Rokach, Maimon & Shmueli, 2023).

All in all, according to Rokach, Maimon & Shmueli (2023) an autoencoder model pursues the objective of minimization of the objective function represented in 2.1.

$$\arg\min_{A,B} [E[\Delta(x, B \circ A(x))] + \lambda \Sigma_i |a_i|] \tag{2.1}$$

Furthermore, a convolutional autoencoder employs convolutional layers instead of the fully connected layers that are used in a simple autoencoder. In convolutional autoencoders, the size of the input layer is the same size as the output layer likewise in simple autoencoders, yet encoding network uses convolutional layers instead of fully connected layers, and the decoding network changes to transposed convolutional layers (Zhang, 2018).

2.2.3 Convolutional Neural Networks (CNNs)

Due to its significant success, the convolutional neural network (CNN) is one of the most frequently used algorithms in the field of deep learning. CNNs have made groundbreaking progress in areas like face recognition, medical diagnosis, and autonomous driving. The CNN is kind of a neural network that can extract features with its convolutional layers. Compared to traditional machine learning methods, CNNs do not rely on manual feature extraction (Li et al., 2021; Lindeberg, 2012; Ahonen, Hadid & Pietikainen, 2006).

Additionally, according to the findings by Li et al. (2021), CNNs hold the following advantages compared to fully connected networks:

1. CNNs demonstrate efficiency in trimming down model parameters and accelerating convergence because each neuron is linked to a smaller portion of neurons instead of being connected to all neurons of the previous layer.
2. Weights can be shared among a group of connections which results in condensing the parameter set further.
3. Dimension reduction takes place using downsampling which is facilitated by utilizing a pooling layer to compress input size. Thereby, the volume of the data shrinks, while important associations are preserved. Beyond that, by eschewing insignificant features the parameter set is reduced.

A CNN architecture is comprised of convolutional, pooling, and fully-connected layers. The convolutional layer which is made up of several feature maps, pursues the objective of acquiring input feature representations. Within the feature map, individual neurons are interlinked to a neighborhood in the preceding layer, which is regarded as the receptive field. In generating future maps, the input data is convoluted by learned kernels, and then a nonlinear activation function is applied to all elements. Tanh, sigmoid, and ReLU are activation functions that are ubiquitously utilized in CNNs (Gu et al., 2018; LeCun et al., 2012; Nair & Hinton, 2010).

2.2.4 Long Short-Term Memory

Lipton, Berkowitz & Elkan (2015) state that recurrent neural networks (RNNs) are efficient at processing sequential data such as time series, audio, and natural language. Similarly, the study by Sutskever, Martens & Hinton (2011) states that RNNs are suitable for fulfilling sequential tasks because they have high-dimensional hidden states with a non-linear nature which enables them to recall previous information. Figure 2.8 represents the mechanism of RNNs.

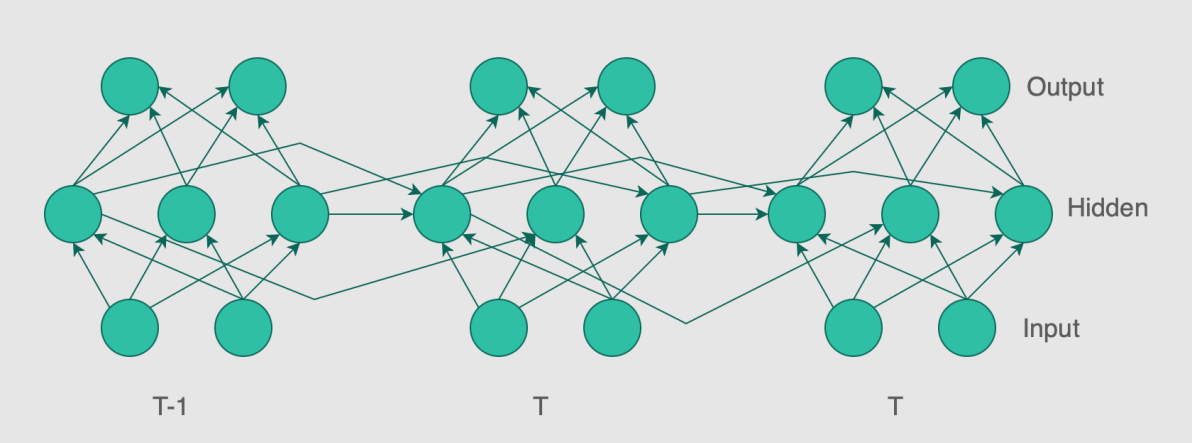

Figure 2.8: Recurrent Neural Network Architecture - Sutskever, Martens & Hinton (2011)

The architecture of an RNN is akin to that of a multi-layer perceptron. Yet, RNNs differ from multi-layer perceptrons in that RNNs allow connections among hidden layers with time delay. These connections enable RNNs to preserve information about the past. Hence, RNNs can detect temporal correlations among events that are far-fetched from each other. On top of that, RNNs are regarded as a simple and capable model though, they are difficult to train appropriately in real-life scenarios. The vanishing gradient and the exploding gradient problems constitute this obstacle. (Pascanu, Mikolov & Bengio, 2013; Bengio, Simard & Frasconi, 1994).

Long short-term memory (LSTM) is one of the most capable sorts of RNN. LSTM replaces conventional artificial neurons in the hidden layer by enforcing the memory cell. The memory cell is capable of linking distant memories, which enables the LSTM network to learn patterns in the data vibrantly with high prediction accuracy (Hochreiter & Schmidhuber, 1997; Chen, Zhou & Dai, 2015).

To summarize, LSTM can cope with long-term reliances in sequential tasks, while conventional RNNs encounter difficulties with long-term dependencies due to vanishing or exploding gradients problems. By contrast to traditional RNNs, LSTM utilizes gates to grasp information flow to tackle the vanishing or exploding gradients problem. Thus, LSTM networks can retain information over long periods (Gers, Schmidhuber & Cummins, 2000). Figure 2.9 depicts the architecture of an LSTM cell.

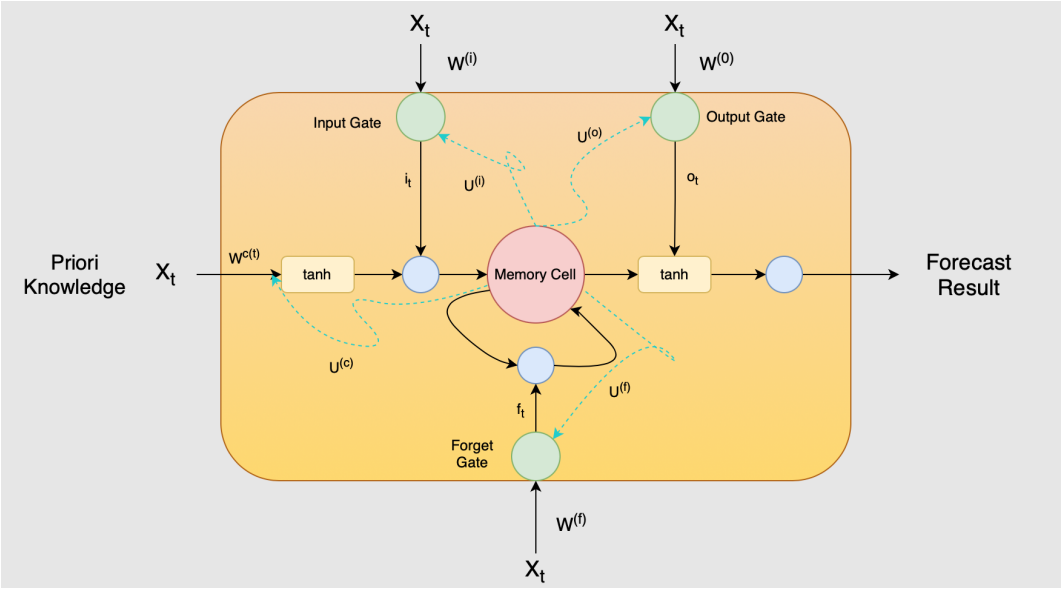

Figure 2.9: LSTM Architecture - Zhao et al. (2017)

An LSTM layer encompasses an input gate, a forget gate, and an output gate. Those gates control information inflow and outflow through the memory cell. The input gate controls which new information should be stored in the memory cell. It receives the previous hidden state and the actual input, then it generates a value between 0 and 1 to the memory cell. In case the value equals zero, the information is dismissed. Otherwise, the information is stored. The forget gate deals with which information to eschew from the memory cell. Similarly, it takes in the previous state and the current state, and then it yields a value between 0 and 1. If the value is zero, then the information is deleted, and it is preserved otherwise. The output gate determines which information should be released from the memory cell. It does so by

considering the previous hidden state and the current input. Hence, it produces an output that reflects the state of the memory cell (Chung et al., 2014).

2.2.5 Underfitting and Overfitting

Underfitting manifests when the machine learning model can not acquire an adequately low error value on the training data. Overfitting happens when a large disparity takes place between training error and test error (Goodfellow, Bengio & Courville, 2016).

Thereby, underfitting stems from the inability to recognize important associations in the training data, which leads to low accuracy and high values of loss. One reason that constitutes the occurrence of underfitting is the oversimplicity of neural network architectures. In other words, when neural networks have a few hidden layers and insufficient training parameters they may not learn complicated properties and relations in the data. Another underlying reason for underfitting is insufficient training time which undermines neural networks' ability to learn relevant patterns in the data. Furthermore, overfitting is referred to as neural networks' over-reliance on the training set. In that case, the model performs well on the training data, however, the model's performance on the validation set is impoverished. What it boils down to is that neural networks can overly concentrate on some particular features of the training data and disregard frequently observed ones, in case an over-complicated model architecture is not fed with sufficient data. In addition to that, if the training set represents characteristics that are different than validation and test sets, it can lead to overfitting (Zhang, Zhang & Jiang, 2019; Smith, 2018, Hinton et al., 2012).

3 State of the Art

This chapter discusses the current studies and research regarding porosity detection and signal processing. In providing an overview of existing research, and methodologies we aim to facilitate a methodology to answer our research question which revolves around porosity detection in additive manufacturing using machine learning. First, we will focus on porosity, and deformity detection in Section 3.1. Then, we will touch upon signal processing and data compression using autoencoders in Section 3.2.

3.1 Porosity and Deformity Detection using Machine Learning

To begin with, the study by Smoqi et al. (2022) uses the methodology of extracting only four features from an imaging pyrometer like temperature and melt pool shape. And, it uses these features to predict porosity by using simple machine-learning models. According to Smoqi et al. (2022) this methodology holds the following advantages:

1. Features used in this study are physically intuitive and easy to handle. Interpretability and computational simplicity enable quick training of models with small data sets.
2. Utilization of computationally simple machine learning approaches instead of deep learning algorithms contributes to fast processing of feedback correction

The article by Nalajam & V (2021) delves into microstructural porosity segmentation using machine learning methods. In this study, microscopic images are used for the purpose of detecting pores. These images are processed in order to extract texture features. Furthermore, k-means clustering, support vector machines, and random forest techniques are used to segment the images. The study makes a meaningful impact because it provides input in the following areas:

1. Forecasting pores in microstructural images with small dataset using machine learning methods and feature extraction
2. Achivening high model accuracy in detecting pores
3. Adequet segmentation of pores using machine learning techniques

The paper by Satterlee et al. (2022) revolves around the concept of comparison of machine learning methods to automate the classification of pores in additive-manufactured parts. For the purpose of training machine learning algorithms Satterlee et al. (2022) created a dataset made up of pores and non-pores as images. The study uses an intensity threshold and gradient filter to choose porous regions in visual data. The selected regions are then labeled as porous and non-porous. For feature extractions from image data, the study evaluates two approaches. The first approach is to build specific methods to extract features by using domain knowledge. The second one delves into the utilization of an automated technique. Moreover, the research applies six machine-learning algorithms, namely support vector machine, boost trees, decision trees, a single layer-neural network, binary linear classifier, and k-nearest neighbor. Beyond that, Satterlee et al. (2022) states that the elimination of features that do not provide support in prediction pores is crucial to improve prediction quality, accelerate computation time, and evade overfitting. to. Hence, the researchers performed feature extraction through a stochastic coordinate descent algorithm. Figure 3.1 summarizes experimental results of Satterlee et al. (2022).

ML Algorithm		Feature reduction method				
	KNN	DT	Linear	SVM	ANOVA	None
Single Layer ANN	**85.8%**	84.4%	78.1%	85.6%	84.6%	84.1%
KNN	84.5%	83.6%	77.2%	84.3%	83.2%	78.1%
BT	85.3%	85.6%	77.9%	85.2%	85.2%	**85.8%**
DT	**79.9%**	79.8%	70.5%	**79.9%**	79.4%	79.3%
Linear	68.5%	64.8%	**74.2%**	68.4%	68.5%	68.4%
SVM	85.0%	85.1%	77.5%	**85.2%**	81.2%	74.3%
#Features	18	10	6	15	24	51

Figure 3.1: Accuracy of training sets and ML methods with the number of features - Satterlee et al. (2022)

The study by Khanzadeh et al. (2018) focuses on using a thermal monitoring system to acquire melt pool signal that is labeled as either porous or non-porous. The article mentions the use of supervised learning methods for capturing patterns in melt pool images. The research is centered on the utilization of support vector machines, decision trees, k-nearest neighbors, and discriminant analysis to detect porosity. According to the findings of the research, k-nearest neighbor provides the highest rate of recall. The decision tree yields the lowest value of mistakenly detecting normal melt pools as pores. Additionally, Khanzadeh et al. (2018) highlights that supervised learning methods showcase high efficiency in identifying anomalies in additive-manufactured parts.

The research by Smoqi et al. (2022) hovers around the topic of optimization of forecasting porosity using data-driven AI classification methods. This study is centralized on implementing random forest, support vector machines, k-nearest neighbors, and extreme gradient boosting (XGBosst) methods. To gauge the classification quality, the study uses accuracy and recall as performance metrics. Smoqi et al. (2022) concludes that extraction of topological features rather than the common ones diminishes accuracy. Because, the presence of pores in parts is associated with absolute values received by the recorded signals, not with their shapes.

The analysis by Eschner et al. (2019) deals with acoustic process monitoring using neural networks. Eschner et al. mentions that recorded signals undergo feature extraction for machine learning, which ends up in a manageable input vector. To preprocess the input vector, raw signals are processed using Fast Fourier transformation. Then, the transformed acoustic signals, and the pressure parameters are then integrated into a dataset. These datasets are to be trained using an artificial neural network, implemented using scikit-learn, and Tensorflow. The complete data is split into a training set (70%) and a test set (30%) to test out repeatability. For cross-validation, the data is split into an 80/20 ratio. To evaluate the model's performance, the study uses F1-Score, precision, and recall. In the article, it is stated that stochastic gradient descent (SGD) gave better results compared to the Adam optimizer. In addition to that, the choice of activation function is ReLU, which provided better results compared to Tanh. Furthermore, the study investigates one, two, and three-layered models performance-wise. In accordance with the findings of the study, the tree-layered model outperforms the others, yet it is more resource-intensive to train this model.

The survey by Wu, Wei & Terpenny (2018) engages in the prediction of surface roughness in additive manufacturing using machine learning, namely random forest. In the framework of the study of Wu, Wei & Terpenny (2018) five features were extracted from the raw data. These features are maximum, minimum, mean, median, and standard deviation. The random

forest model is trained on these features. To assess the model's capability 10-cross validation method is used. According to the experimental findings of the study, the machine learning model showcases high capability in predicting the surface roughness of the printed parts.

The article by Caggiano et al. (2019) focuses on image processing to identify defects in additive manufacturing using machine learning. Within the context of this work, a bi-stream deep convolutional neural network (DCNN) is developed to recognize deformities. On top of that, the article emphasizes that the success of DCNN in various manufacturing settings was proven by Wang et al. (2018), Weimer, Scholz-Reiter & Shpitalni (2016). Caggiano et al. uses DCNN to detect deformities in part images. All in all, according to the experimental results of Caggiano et al. (2019) the method achieved an accuracy of 94%.

To detect welding defects in steel plates Kothari (2018) uses machine learning and computer vision algorithms. The study deals with image data and uses a U-Net model to perform predictive analysis. Kothari (2018) follows the following steps to detect the welding deformities:

1. Segment the images
2. Illustrate severity of deformities with colors
3. Use image moments to gauge the severity
4. Train the U-Net model

Kothari (2018) states that the U-Net model receives input data of dimension 512x512x3. Thereby, the input data was reshaped into that dimension. Next, the image was normalized through division by 255, which accelerates the computation.

The research by Satterlee et al. (2023) is centered on using machine learning to detect pores in images of metal parts manufactured through binder jetting. The research adopts the methodology of using CNNs to classify images. First, a ResNet50 deep CNN is trained within the frame of the study. However, the model can not achieve the desired performance level. Consequently, the study seeks out another method that can demonstrate a higher performance. An extension of CNN named R-CNN proposes a solution. It is stated by Ren et al. (2017) that there are three types of R-CNN. These are R-CNN, Fast R-CNN, Faster R-CNN. Satterlee et al. (2022) says that Faster R-CNN differs from the other models in that image data is fed through the network only once. By contrast, R-CNN entails a proposal network. In R-CNN each region is extracted and independently classified, which leads to longer training time and lower performance.

The scientific inquiry by Cheepu (2023) sheds light on the prediction of defect characteristics in additive manufacturing using machine learning. The article delves into using linear regression, decision trees, and random forest models for identifying deformities through processing voltage signal data. To facilitate this, the data is split 80% to 20% as a training and testing set. Moreover, root mean square error (RMSE) and R^2 are used to assess the fitness of the machine learning models. The findings by Cheepu (2023) report that the linear regression model can not predict deformities accurately. Furthermore, the decision tree model is trained contingent upon the classification of the deformities, features of the deformities, and fluctuation in signals. The model achieves an accuracy of 99.13%. What is more, the random forest model outperforms the other models with a prediction accuracy of 99.78%. All in all, pores and surface defects are detected adequately by the model. Hence, in accordance with the research outcomes by Cheepu (2023) the model is suitable to be employed for the detection of deformities in real-time processes.

3.2 Data Compression and Signal Processing using Autoencoders

The study by Kuester, Gross & Middelmann (2021) delves into hyperspectral data compression through a 1D convolutional autoencoder. The paper states that the electromagnetic spectrum of an area is measured by hyperspectral sensors. On top of that, according to the study machine learning algorithms demonstrate success in signal processing. Additionally, autoencoder models are highly regarded in the domain of hyperspectral image processing. So, the research by Kuester, Gross & Middelmann (2021) delves into the analysis of the reconstruction accuracy of spectral dimension via 1D-convolutional autoencoder. Figure 3.2 represents the autoencoder model used by Kuester, Gross & Middelmann (2021).

	Layer Type	Activation Function	Kernel Size
	-	-	-
	1D Zero Padding	-	-
	1D Convolutional	Leaky ReLU	11
	1D MaxPooling	-	2
	1D Convolutional	Leaky ReLU	11
	1D MaxPooling	-	2
	1D Convolutional	Leaky ReLU	9
ENCODER	1D Convolutional	Leaky ReLU	7
	1D Convolutional	Leaky ReLU	7
	1D Convolutional	Leaky ReLU	9
	1D Upsampling	-	2
	1D Convolutional	Leaky ReLU	11
	1D Upsampling	-	2
	1D Convolutional	Sigmoid	11
DECODER	Cropping 1D	-	-

Figure 3.2: Convolutional Autoencoder Architecture - Kuester, Gross & Middelmann (2021)

The research by Proteau et al. (2020) thematizes reducing the dimensionality of industrial data by using a variational autoencoder approach. The study highlights that in variational autoencoder applications, each element sampled from the latent space can be represented in accordance with the Equation 3.1:

$$z_i = \mu_i + \sigma_i \cdot \epsilon \tag{3.1}$$

Proteau et al. (2020) states that variational autoencoders also differ from traditional autoencoders with their loss function. Equation 3.2 defines the loss function of the variational autoencoder.

$$\mathcal{L} = E_{q\phi(z|x)} \log \left[P_\theta \left(x|z\right)\right] - KL \left[q_\phi \left(z|x\right) \parallel P(z)\right] \tag{3.2}$$

To sum up, Proteau et al. (2020) expresses that it aims to present the industrial data set that it uses throughout the study. It pursues the objective of amplifying different features of which the input data is made up, and finally, it intends to demonstrate variational autoencoder models along with a classifier.

Sakurada & Yairi (2014) discusses dimensionality reduction using autoencoders for anomaly detection. In accordance with the article by Chandola, Banerjee & Kumar (2009), Sakurada & Yairi remarks that dimensionality reduction regarding anomaly detection relies on the assumption that data instances in the input data are correlated with each other and they can be represented in a lower dimensional subspace wherein normal instances are remarkably distinguished from anomalous instances. Sakurada & Yairi (2014) works toward compressing the input data into a lower dimensional latent subspace, and reconstruct $(x^{(\hat{1})}, x^{(\hat{2})}, \ldots, x^{(\hat{m})})$ by which the reconstruction loss represented in 3.3 is minimized.

$$\sqrt{\sum_{j=1}^{n} (x_j^{(i)} - \hat{x}_j^{(i)})^2} \tag{3.3}$$

Beyond that Sakurada & Yairi (2014) notes that the autoencoder model minimizes the objective function shown in Equation 3.4 that is parametrized by W and b. The objective function is composed of an error term and a regularization term. The regularization parameter λ indicates the scope of regularization, wherein n_l represents the number of layers and s_l parametrizes the number of neurons in the corresponding layer.

$$J(W,b) = \frac{1}{m}\sum_{i=1}^{m}(\frac{1}{2}\|x^{(i)} - \hat{x}^{(i)}\|^2) + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_l+1}(W_{ij}^{(l)})^2 \tag{3.4}$$

Sakurada & Yairi (2014) sets up an experimental context in which linear PCA, kernel PCA, autoencoder, and denoising autoencoder are tested out and compared with respect to reconstruction loss in the domain of dimensionality reduction. The methods are trained on two data sets, namely artificial data and real data.

Ren et al. (2022) touches upon quality monitoring in additive manufacturing using unsupervised deep learning. The study sums up its proposed LSTM-autoencoder model as follows:

1. A layer with a dimension of 5x410 is utilized as an input layer.
2. Two LSTM layers are imposed after the input layer to address the sequential nature of the data.
3. A repeat vector reproduces the encoded data. Additionally, two more LSTM layers are enforced in the opposite order of the encoder model.
4. In the last LSTM layer sigmoid activation function is incorporated to squeeze the values in the range of 0 and 1. Furthermore, the tanh activation function is used in other LSTM layers. The model aims to minimize the reconstruction loss. In other words, it pursues the minimization of the difference between input data and the reproduced output.

4 Own Approach

This chapter discusses the approach to address the research questions posed by this study, which delves into the use of autoencoders, LSTM, and CNNs, to process signal data. This approach is centered around using the power of deep learning to retrieve meaningful associations and patterns from complex signal data. Furthermore, this methodology aligns with novel techniques that are highlighted in Chapter 3. In summary, this chapter outlines the theoretical context of the methodology and deals with the logic of using autoencoders and artificial neural networks in this specific domain.

To summarize, Section 4.1 discusses data collection, data characteristics, and the problem. After analysis of the data, the methodology that revolves around feature engineering and machine learning methods is highlighted in Section 4.2.

4.1 Exploring the Data

This section discusses how the data was collected and the characteristics that it represents. In the first step, we will touch upon the data collection process. Then, the research question posed by this study will be introduced. Additionally, the methodology that will be employed to answer this question will be explained.

It is important to explore the data to get insights into its main properties. For this purpose, we will refer to the statistical attributes of the data. Additionally, we will utilize visualization methods to provide clearer insights into the data and labels. These steps are significant to envision the construction of a data processing and machine-learning pipeline. After providing information on data characteristics and labels, we will discuss the data preparation process and machine learning models.

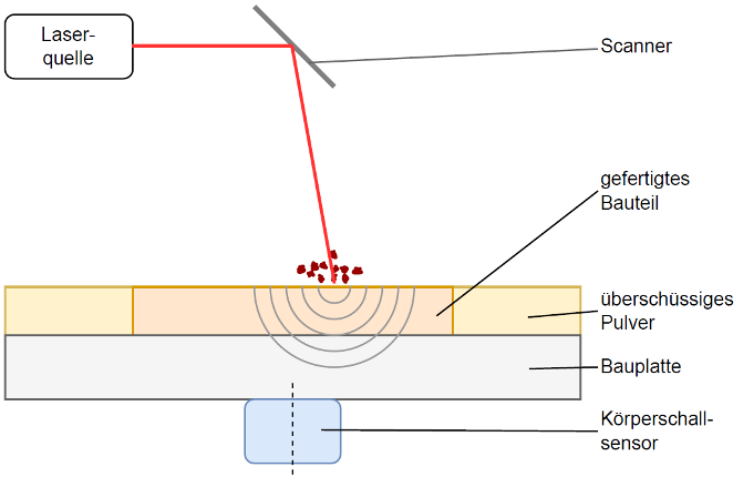

Figure 4.1: Collecting Signal Data

Figure 4.1 shows the process of signal data acquisition in the framework of this study. According to the figure, a laser source sends beams onto an additive-manufactured part. When these beams collide with the part, they generate acoustic signals within that part. The signals are recorded by a sensor placed underneath the built plate. As an additive-manufactured part is constructed through the layer-by-layer method, these signals are used as indicators to detect whether there are pores in the layers that compose the part. Figure 4.2 depicts a spatially resolved representation of pores.

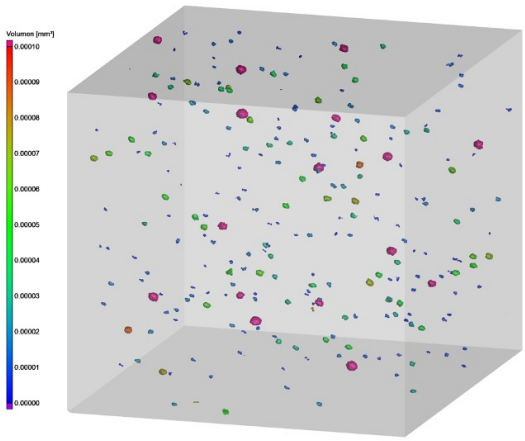

Figure 4.2: Spatially Resolved Representation of Pores

After the signals are recorded by the sensor, they are converted from analog to digital form by an amplifier. Then, the digitized data are transferred to a measuring computer.

Our data is stored in h5 files. In Chapter 2, it is mentioned that additive-manufactured parts are constructed layer by layer. Hence, each HDF5 file corresponds to one layer that constitutes the product. Consequently, all HDF5 files must be converted into another type of data structure whereby one can conduct arithmetic operations. To address this, all files are converted into NumPy arrays, as the NumPy library in Python enables us to conduct arithmetic operations with arrays. A NumPy array is a data structure that can store n-dimensional arrays.(Oliphant et al., 2006; Harris et al., 2020).

Besides that, each layer is associated with a label that represents porosity. The labels are composed of float and scalar values. They are stored in a CSV file. For the purpose of computability, it needs be to converted into a NumPy array. After all files had been converted into NumPy arrays, a data frame was created by using pandas to illustrate our features and labels in order to solidify our understanding of the dataset. Figure 4.3 represents the resulting data.

	id	Data	Lenght	Porosity
0	1	[8357728, 8356380, 8354842, 8348277, 8349389, ...	2647040	4.61
1	2	[8356094, 8357054, 8358442, 8351578, 8353312, ...	2631680	4.56
2	3	[8355636, 8355458, 8354434, 8342865, 8344323, ...	2672640	4.18
3	4	[8351555, 8350919, 8351099, 8364344, 8363224, ...	2641920	3.54
4	5	[8342533, 8344145, 8345981, 8349178, 8349924, ...	2647040	2.63
...	...	...	...	...
555	556	[8359072, 8360424, 8361844, 8351455, 8349789, ...	2657280	1.95
556	557	[8353729, 8354439, 8354487, 8358317, 8356479, ...	2672640	1.82
557	558	[8351199, 8353325, 8355619, 8359743, 8359185, ...	2631680	2.19
558	559	[8347116, 8348722, 8350138, 8343370, 8343968, ...	2667520	2.99
559	560	[8357172, 8356206, 8354810, 8351744, 8349810, ...	2641920	3.10

560 rows × 4 columns

Figure 4.3: Data

In Figure 4.3, the "Data" column refers to acoustic signals, namely our features. The futures are float values that represent sequential association. The "Length" column shows the length of each feature. Finally, the last column "Porosity" represents our target, in other words, the labels. In summary, we have 560 features of different lengths, and each feature is associated with a float label. By using NumPy, we can readily find out the minimum and maximum values of each column. The maximum value of the "Lenght" column is 2698240 while the minimum value is 2611200. For the column "Porosity" the maximum value is 7.54, and the minimum value is 0.71. Furthermore, the column "Porosity" has a variance of 0.74, a median of 2.23, and a mean of 2.34. To investigate the dataset further, one thing we can do is visualize our features and labels. Figure 4.4 shows the distribution of the features in accordance with the labels.

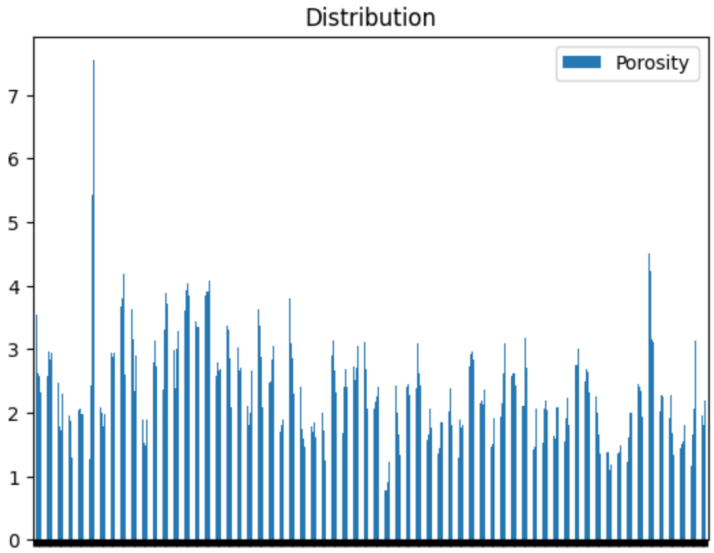

Figure 4.4: Distribution of the Dataset

In processing the data represented in Figure 4.3, one of the biggest obstacles is the volume of the data. To visualize the data and draw inferences from it, the volume of the data ought to be reduced. Principal Component Analysis(PCA) is one of the widely used methods to process data into a lower dimensionality space. There are tradeoffs in dimension reduction. PCA assumes a linear relationship and the maximal variance in the data. In addition, uniform manifold approximation and projection for dimension reduction (UMAP) is a novel method for dimension reduction. UMAP is not computationally restricted to dimensions. Consequently, it is considered a general dimension reduction method. Beyond that, UMAP can handle nonlinear relations and preserve local associations like clusters in the reduced data (McInnes, Healy & Melville, 2018; Diaz-Papkovich, Anderson-Trocmé & Gravel, 2021).

All these properties of UMAP make it suitable for reducing data to generate visualizations. By using UMAP, we can reduce the data to a lower dimension for visualization purposes. Figure 4.5 demonstrates the reduced version of the data depicted in Figure 4.3 into UMAP dimensions.

After compressing our features using UMAP, we can depict the data in two UMAP features. The number of UMAP features is a hyperparameter to be tuned. In this example, the data is reduced into a two-dimensional UMAP space for visualization purposes.

In the raw data, the feature with the minimum size has a length of 2611200. As there are 560 features, we would end up with data with the shape of 560x2611200, which means that the data is composed of 560 features with each containing 2611200 elements. In the next step, we can visualize the data set in accordance with the labels to analyze the distribution of the labels through the data set.

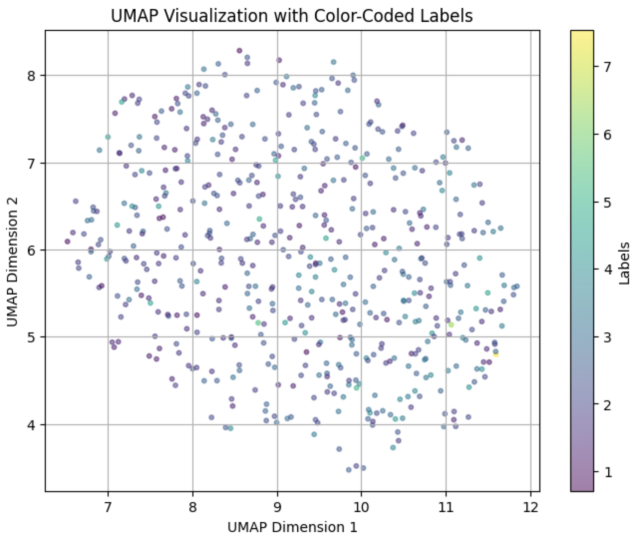

Figure 4.5: UMAP Distribution

In Figure 4.5, the scale of the labels is depicted with colors. The figure indicates that purple and green tones are dominant over others. This implies that the value of the labels is mainly distributed between 2 and 3. Furthermore, according to Figure 4.5 labels between the values of 6 and 7 demonstrate the minority class, as yellow tones are rarely observed. In addition to this, Figure 4.6 depicts the histogram of the labels to provide a clearer point of view.

Visualization of the histogram of the labels is important to obtain insights into the distribution of them. As we have continuous float labels, they need to be converted into binary values when we tackle with classification of porous layers. In terms of setting adequate threshold values to convert the continuous labels into binary values, and to check out if the data is imbalanced Figure 4.6 provides us guidance.

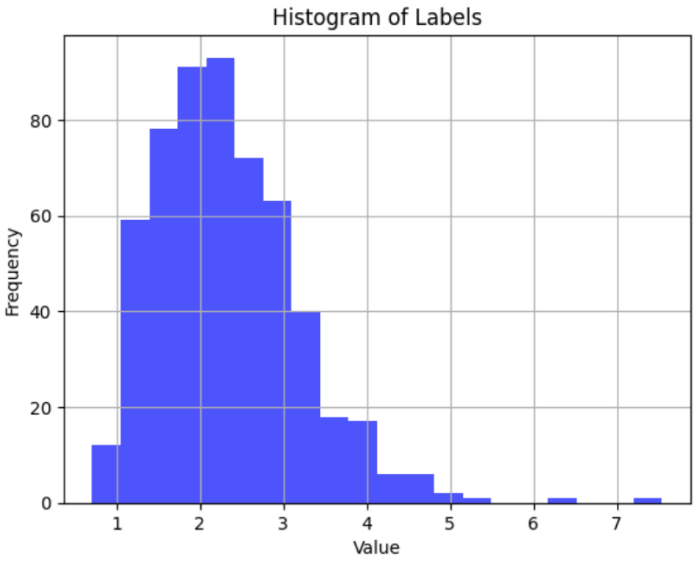

Figure 4.6: Histogram of Labels

Figure 4.6 indicates that labels with a value smaller than 1 or greater than 4 represent the minority. We may have to take this into account when dealing with classification problems. So far, the nature and the characteristics of the dataset and the corresponding labels are explained. The research question revolves around predicting the labels, namely porosity which is depicted in float numbers, using acoustic signals. As discussed in Chapter 3, machine learning and deep learning propose solutions to process signal data to predict porosity. To facilitate this approach, a path to resolution is needed. The next section discusses the methodology to be employed to address the research question.

4.2 Methodology

In Section 4.1, we delved into the nature and the characteristics of the dataset. Based on this, we should construct a machine learning pipeline to process the dataset to perform predictions. The solution model proposed by this study is depicted in Figure 4.7.

First of all raw data should be preprocessed and feature-engineered to make it digestible for machine learning models. After we trim the dataset to equalize the length of each layer so that it can be processed by deep learning models, we are left with data with the shape of 560x2611200. Thus, this makes 1462272000 data points in total which represents a huge dataset to be processed. To preprocess the dataset methods proposed by Pedregosa et al. (2011), Buitinck et al. (2013) are to be used.

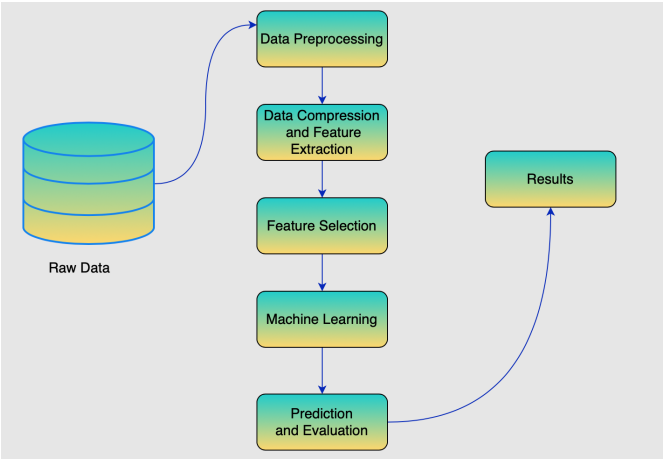

Figure 4.7: Methodology

Buitinck et al. (2013) states that scikit-learn is an open-source machine-learning library developed on top of Python. The library provides machine-learning concepts in the framework of the Python programming language that can be utilized by non-machine-learning experts in scientific settings. Scikit-learn encompasses model selection and evaluation instruments, machine-learning algorithms, and preprocessing methods. According to Buitinck et al. (2013) scikit-learn is structured on the following principles:

- Consistency
- Inspection
- Non-proliferation of class
- Composition
- Sensible defaults

Figure 4.8 depicts the preprocessing of a dataset using a standard scaler proposed by the scikit-learn library. Before preprocessing the dataset, it has a mean of 15.96 and a standard deviation of 30.69. After preprocessing the data has a mean of 0, and a standard deviation of 0.99. This means that the standard scaler by scikit-learn transforms the data so that it has a standard normal distribution. Hence, the data is centered around 0 and with a standard deviation of 1. For instance, we can also use MinMaxScaler offered by scikit-learn to scale data between -1 and 1. Note that the data depicted in Figure 4.8 is not our original dataset, it is randomly generated to demonstrate how scaling and preprocessing of a dataset functions.

```python
from sklearn import preprocessing
import numpy as np

X_train = np.array([[ 8., -1.,  20.],
                    [ 15.,  0.,  0.7],
                    [ 100.,  7., -6.]])


print(f"Mean of  X_train is : {X_train.mean()}")
print(f"Standard deviation of X_train is : {X_train.std()}")

scaler = preprocessing.StandardScaler().fit(X_train)
print("\n")

X_scaled = scaler.transform(X_train)
print(f"Scaled data: {X_scaled}")
print("\n")
print(f"Mean of X_scaled is : {X_scaled.mean()}")

print(f"Standard deviation of X_scaled is : {X_scaled.std()}")
```

```
Mean of  X_train is : 15.966666666666665
Standard deviation of X_train is : 30.697593101450515


Scaled data: [[-0.78915157 -0.84292723  1.36997499]
 [-0.62175578 -0.56195149 -0.38105265]
 [ 1.41090735  1.40487872 -0.98892234]]


Mean of X_scaled is : 1.2335811384723961e-17
Standard deviation of X_scaled is : 0.9999999999999999
```

Figure 4.8: Preprocessing using scikit-learn

Our dataset is too large to handle by a neural network. Therefore, a representative subset is to be chosen. Then, this subset is to be compressed by an autoencoder model for feature extraction. Simple, recurrent, convolutional, sparse, and simple feedforward autoencoder models will be tested. And then we will go ahead with the model that gives the best performance metrics. In the next step, the encoded data is to be used as input for an LSTM, CNN, and a multi-layer perceptron model.

In the first step, data should be compressed. In this study, data compression and feature extraction of signal data revolve around autoencoders. Before training autoencoder models, a subset of the data that has the shape of 560,10000 is chosen. This means that the dataset is composed of 560 arrays (each array refers to one layer) each containing 10000 elements. To compress this dataset, autoencoder models to be tested are depicted in Table 4.3, 4.2, 4.3, and 4.4.

Layers	Explanation
LSTM (encoder)	LSTM layer
Repeat Vector	RepeatVector Layer
LSTM (decoder)	LSTM layer
Output	Output layer

Table 4.1: Recurrent Autoencoder Architecture

Layers	Activation
Encoder	
Conv1D (64 filters, kernel 3)	ReLU
MaxPool1D (kernel 2)	
Conv1D (32 filters, kernel 3)	ReLU
MaxPool1D (kernel 2)	
Conv1D (128 filters, kernel 3)	ReLU
Decoder	
ConvTranspose1D	ReLU
ConvTranspose1D	ReLU
ConvTranspose1D	ReLU

Table 4.2: Convolutional Autoencoder Architecture

Layers	Explanation
Encoder Model	
Dense (128 units, ReLU)	Hidden layer
Dense (64 units, ReLU, L1 Regularization)	Bottleneck
Decoder Model	
Dense (128 units, ReLU	Hidden layer
Dense (linear activation)	Output layer

Table 4.3: Sparse Autoencoder Architecture

Layers	Explanation
Encoder Model	
Dense (64 units, ReLU)	Hidden layer
Decoder Model	
Dense (ReLU)	Hidden layer
Dense (linear activation)	Output layer

Table 4.4: Simple Autoencoder Architecture

Figure 4.9 depicts the input data for the autoencoder model. The input data is an array that is composed of 560 subarrays. Each subarray represents one layer and contains 10000 data points. Using an autoencoder, weaim to reduce the input data through the encoder layer of the autoencoder. The encoder layer of the autoencoder generates encoded data that refers to the representation of the input data in the latent space. The encoding of the input data pursues the objective of extracting features, preventing overfitting, and speeding up computation for further training using another deep learning model for predictive purposes. Figure 4.10 shows the resulting encoding data after training with an autoencoder model.

```
Input data: [[-0.14513085  0.02092341  0.31068091 ... -0.21367783 -0.10354989
   0.13676943]
 [ 0.11677974 -0.1550915  -0.3689003  ... -1.78517599 -1.87677765
  -1.75119605]
 [-0.35604667 -0.55896776 -0.57605044 ...  0.68443506  0.86997138
   1.08085093]
 ...
 [ 0.61922811  0.29612281 -0.10267261 ... -0.49044974 -0.86143778
  -1.04985015]
 [ 0.67584729  0.45654622  0.26277508 ...  0.4399665   0.21435079
   0.08976286]
 [-0.64064243 -0.83134943 -0.86916599 ...  0.92890363  1.02337233
   1.13062259]]
Shape of the input data: (560, 10000)
```

Figure 4.9: Input Data for Autoencoder

The encoded data has the shape of 560,64. This means that the length of each layer is compressed from 10,000 to 64 elements. To summarize, the encoded data will be used as input data for another deep-learning model to predict the pores.

```
Encoded data: [[ 0.          0.          0.6115025  ...  2.182461   10.145544
  13.802161  ]
 [ 0.          0.         10.364564   ...  3.8790486  21.651777
   4.106466  ]
 [ 0.          0.          0.79901433 ... 11.411533   19.60512
  15.882882  ]
 ...
 [11.476929    0.7671157   2.8109496  ...  4.1257696  12.857623
   3.4391353 ]
 [13.025483    0.16663408 13.693355   ...  7.7498016   8.476287
  21.760511  ]
 [ 0.65788126  0.          8.615754   ... 10.092576   25.760952
  11.329406  ]]
Shape of the encoded data: (560, 64)
```

Figure 4.10: Encoded Data

To evaluate the performance of the autoencoder models, the reconstruction loss will be employed as a quantitative metric for performance evaluation. The reconstruction loss indicates whether the reconstructed input, in other words output of the autoencoder model, matches up with the original input data. As the reconstructed input is generated from the encoded data, the reconstruction loss shows if the pattern in the original input data is contained in the encoded data.

This study focuses on two machine learning tasks, namely predicting the float labels and binary classification. To predict the float values, deep learning models depicted in Table 4.5 4.6, and 4.7 will be utilized. To monitor the fitness of the models during training mean squared error(MSE) will be used. After training, the models will be evaluated on a test set based on MSE and R^2. To split the dataset, train-test-split by scikit-learn will be used.

Layers	Explanation
Conv	Convolutional layer(1, 32, kernel-size=(3,), stride=(1,)
ReLU	Activation
Maxpool	Pooling layer MaxPool1d(kernel-size=2, stride=2, padding=0)
Conv	Convolutional layer Conv1d(32, 64, kernel-size=(3,), stride=(1,))
ReLU	Activation
Maxpool	Pooling layer, MaxPool1d(kernel-size=2, stride=2, padding=0)
Linear	Fully connected layer
ReLU	Activation
Linear	Fully connected layer
ReLU	Activation
Linear	Output layer

Table 4.5: CNN Model

Layers	Explanation
Linear	Fully connected layer
ReLU	Activation
Linear	Fully connected layer
ReLU	Activation
Linear	Fully connected layer
ReLU	Activation
Linear	Fully connected layer
ReLU	Activation
Linear	Output layer

Table 4.6: Multi Layer Perceptron

Layer	Explanation	Input Size	Hidden Size
LSTM	LSTM layer	64	50
LSTM	LSTM	50	50
LSTM	LSTM	50	50
LSTM	LSTM	50	50
Fully connected	Linear	50	1

Table 4.7: LSTM Architecture

This study also touches upon the classification of porous layers. To frame the classification problem, a thresholding method is applied to binarize continuous float labels. Different threshold values will be explored, specifically 1.0 and 2.0 to observe their impact on the model's performance. The distribution of classes regarding the labels, after the continuous labels are binarized based on a threshold value of 1.0 is depicted in Figure 4.11.

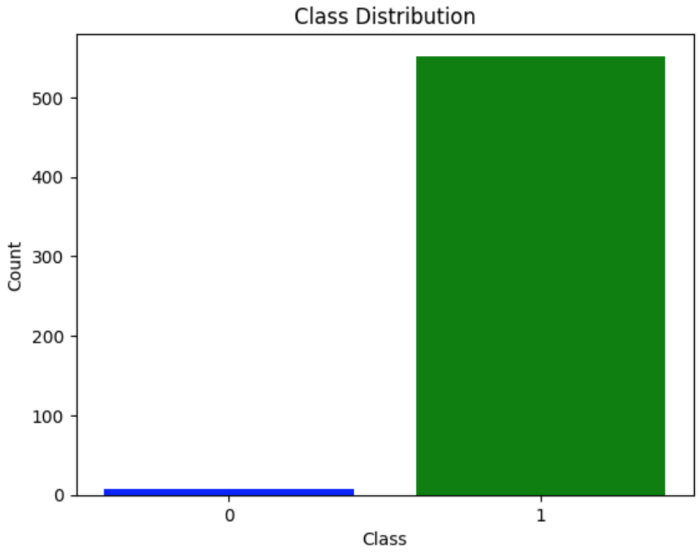

Figure 4.11: Binarized Labels with Threshold Value of 1.0

Figure 4.11 shows that two classes are obtained after the binarization of the labels. Furthermore, the figure highlights that class 0 is heavily outnumbered by class 1.

Class imbalance occurs in some real-world problems when almost all the instances are labeled as one class whereas fewer instances are labeled as the other class. Due to class imbalance, classifiers can heavily focus on the majority class and neglect the minority class. As a consequence of this, machine learning models might yield suboptimal classification performance (Guo et al., 2008).

In case of class imbalance, classifiers tend to over-classify the majority class due to the high likelihood of data points being associated with that class. Hence, data points that are associated with the minority class are misclassified much more frequently than the ones that belong to the majority class. In addition to this, some performance indicators like accuracy may exhibit inflated scores, potentially giving way to deceptive inferences of good performance (Johnson & Khoshgoftaar, 2019).

To tackle the class imbalance problem, Picek et al. (2019) highlights the following resampling methods:

1. **Random Undersampling:** This method undersamples all the classes apart from the least populated one
2. **Random Undersampling with Replacement:** This technique oversamples the minority class by generating data points from the original minority class, with replacement
3. **Synthetic Minority Oversampling Technique:** This approach focuses on oversampling via generating synthetic minority class instances
4. **Synthetic Minority Oversampling Technique with Edited Nearest Neighbor:** This methodology combines oversampling by synthetic minority oversampling technique and data cleaning by edited nearest neighbor

A Python toolbox called imbalanced-learn proposes methods to overcome the class imbalance problem that often occurs in machine learning. Implementation of imbalanced-learn is interfaced with NumPy, SciPy, and scikit-learn. Assume that χ is an imbalanced dataset wherein χ_{min}, and χ_{max} exhibit samples from minority and majority classes respectively. The ratio that balances the dataset is defined as $r_\chi = \frac{\chi_{\text{min}}}{\chi_{\text{maj}}}$. The balancing of the dataset is accomplished by resampling χ into a new dataset χ_{res} so that $r_\chi > r_{\chi_{\text{res}}}$ holds (Lemaître, Nogueira & Aridas, 2017). In the study by Lemaître, Nogueira & Aridas (2017) it is stated that imbalanced-learn proposes four different techniques to cope with the imbalanced dataset problem.

- **Under-sampling:** Undersampling reduces the number of samples in χ_{maj}
- **Over-sampling:** In over-sampling new samples are generated in χ_{min} to facilitate the balancing ratio
- **Combination of over- and under-sampling**: Oversampling can give way to overfitting which can be evaded by enforcement of under-sampling methods (Prati, Batista & Monard, 2009)
- **Ensemble Methods**: Samples from the majority class can be erased if under-sampling is applied. To address this issue, ensemble methods focus on utilizing a significant portion of the samples.

After random oversampling is applied to the dataset by using imbalanced-learn, the distribution of classes takes the form represented in Figure 4.12. The figure indicates that the dataset is balanced after resampling.

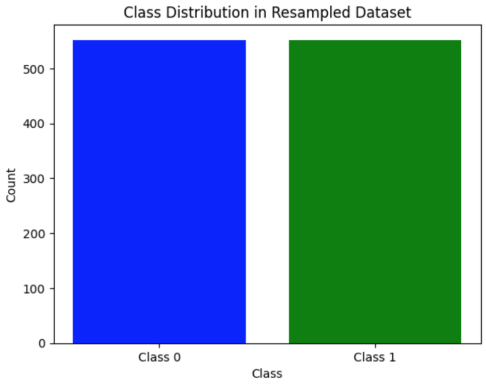

Figure 4.12: Distribution of the Classes After Resampling

In this study, random oversampling, and synthetic minority oversampling techniques (SMOTE) will be applied setting a threshold value of 1.0 and 2.0 to tackle the class imbalance problem. When the threshold is set to 2.0, we obtain the distribution of labels depicted in Figure 4.13.

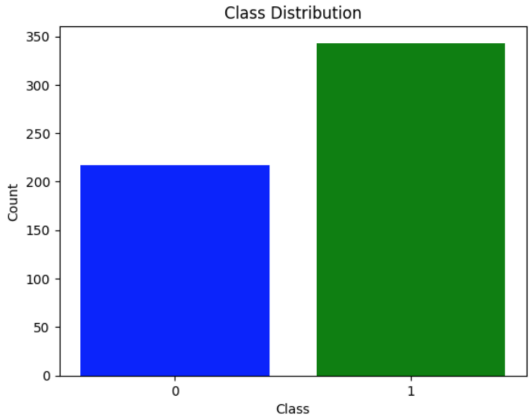

Figure 4.13: Distribution of the Labels with a Threshold Value of 2.0

A multi-layer perceptron, a CNN, and a CNN-LSTM model will be utilized to predict porous layers in the context of a binary classification task. Deep learning models that will be employed to predict the pores are represented in Table 4.8 and Table 4.9. Additionally, the CNN-LSTM model is composed of convolutional, pooling, and LSTM layers. To assess the performance of these models, Equation 4.1, and Equation 4.2 will be utilized as performance metrics. Furthermore, a confusion matrix will be provided.

Table 4.8: Multi Layer Perceptron Architecture Regarding Classification

Layers	Explanation
1	Fully connected
2	ReLU
3	Fully connected
4	ReLU
5	Fully connected
6	ReLU
7	Fully connected
8	ReLU
9	Fully connected
10	Sigmoid

Table 4.9: CNN Architecture Regarding Classification

Layers	Explanation
1	Convolutional layer
2	ReLU (activation)
3	Max pooling layer
4	Convolutional Layer
5	ReLU (activation)
6	Max pooling layer
7	Fully Connected layer
8	ReLU (activation)
9	Fully connected layer
10	ReLU (activation)
11	Fully connected layer
12	Sigmoid

Table 4.10: CNN-LSTM Architecture Regarding Classification

Layers	Explanation
1	Convolutional layer
2	ReLU (activation)
3	Max pooling layer
4	Convolutional Layer
5	ReLU (activation)
6	Max pooling layer
7	LSTM
8	Fully connected layer
9	Sigmoid

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \tag{4.1}$$

$$\text{F1-score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4.2}$$

The models will be trained by using PyTorch. PyTorch is a deep learning framework that enables Python programming style and facilitates computation accelerators like GPUs. PyTorch furnishes NumPy array-like tensors that can be stored either on the CPU or GPU which fosters the acceleration of the training process significantly (Paszke et al., 2019a; Paszke

et al., 2019b). According to Paszke et al. (2019a), PyTorch is structured onto the following design principles:

- **Pythonic:** Python is frequently utilized in data science. PyTorch is a member of this environment.
- **Reseracher-friendly:** PyTorch enables the construction of deep learning models simply and efficiently, encompassing loading data and optimization for machine learning.
- **Deliver pragmatic performance:** PyTorch delivers efficient performance while preserving simplicity.
- **Simplicity over completeness:** PyTorch prefers to opt for a simple but somehow incompleted solution over a comprehensive but difficult-to-sustain design.

To sum up, we explored the data then we constructed a machine-learning pipeline. Additionally, machine learning models regarding regression and classification tasks were introduced. On top of that the imbalanced data problem and solution methods were discussed. In Chapter 5 training results of the machine learning models will be discussed.

5 Results

This chapter revolves around the training results of the deep learning models introduced in Chapter 4. In this chapter, the training results of the autoencoders, deep learning models for regressive analysis, and models for classification tasks will be depicted respectively.

To assess the results of the deep learning models, the training process, namely training loss and validation loss will be visualized. Moreover, the fitness of the models will be evaluated based on performance metrics.

Furthermore, for the evaluation of the autoencoder models, random samples with the same indices from the input data and reconstructed data will be compared in addition to visualization of training and validation loss. In other words, five samples of data with same the randomly generated indices will be selected from the input data and the reconstructed data will be chosen to inspect visually if the reconstructed data looks like the the input data.

First, the training results of the autoencoder models will be shown. Then, the results of the regression and classification will be discussed respectively. As the chapter mainly focuses on providing analytical insights into training results, the results will be evaluated in Chapter 6.

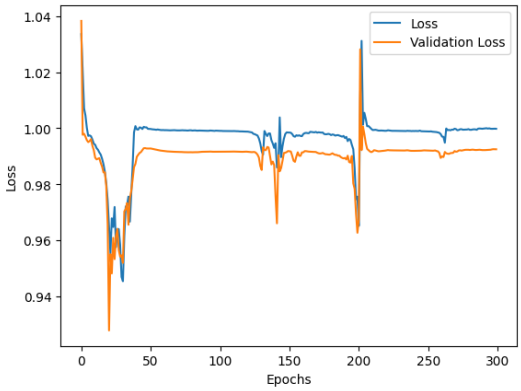

Figure 5.1: Training and Validation Losses during Training of the LSTM Autoencoder Model

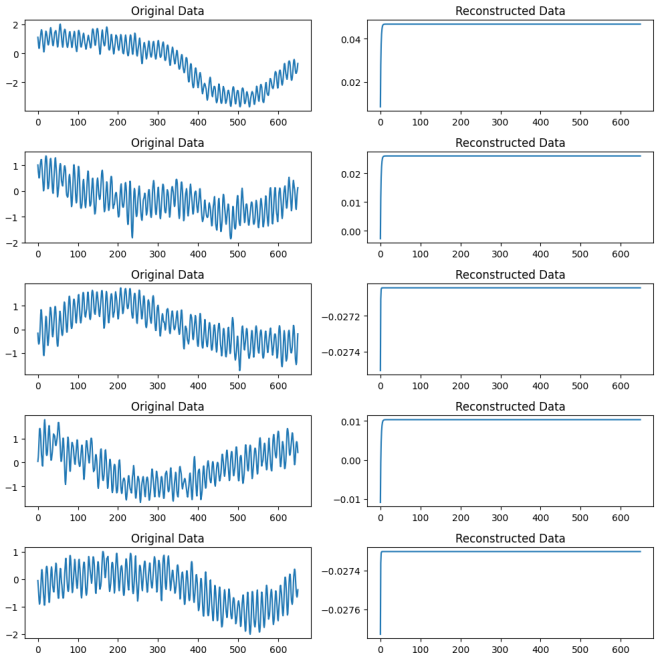

Figure 5.2: Training Results of the LSTM Autoencoder Model

Figure 5.2 indicates that the LSTM autoencoder model can not capture the pattern of the input data. In the figure, samples chosen from the input data do not match up with the ones chosen from the reconstructed data. Furthermore, Figure 5.1 shows that the loss values do not converge. Therefore, the encoded data by the LSTM autoencoder model can not be used to predict the porous layers.

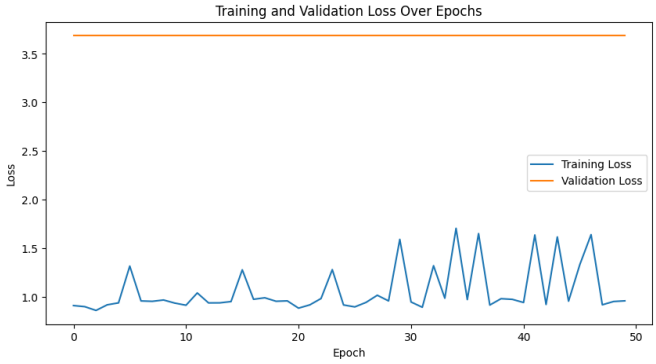

Figure 5.3: Training Process of the Convolutional Autoencoder Model

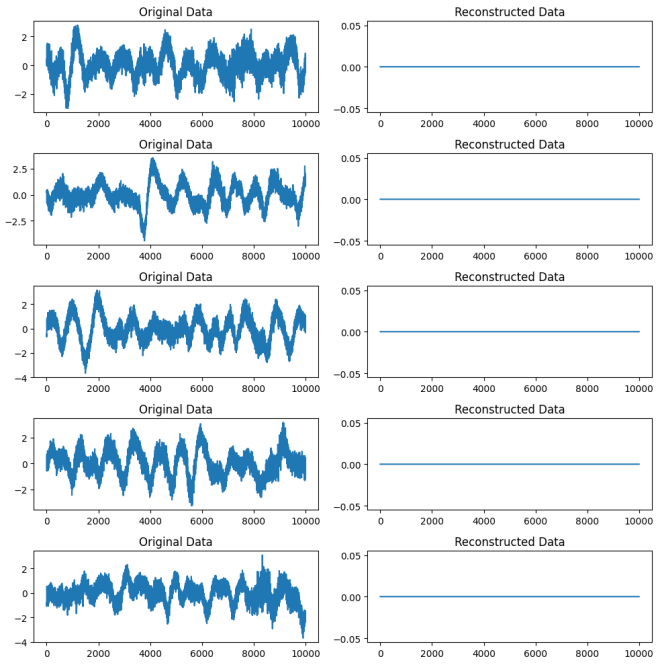

Figure 5.4: Training Results of the Convolutional Autoencoder Model

According to Figure 5.3, validation loss of the convolutional autoencoder model during training does not converge. Moreover, Figure 5.4 depicts that the reconstructed data does not exhibit the pattern of the original data. Consequently, it is not adequate to use the encoded data by the convolutional autoencoder as input to predict the pores.

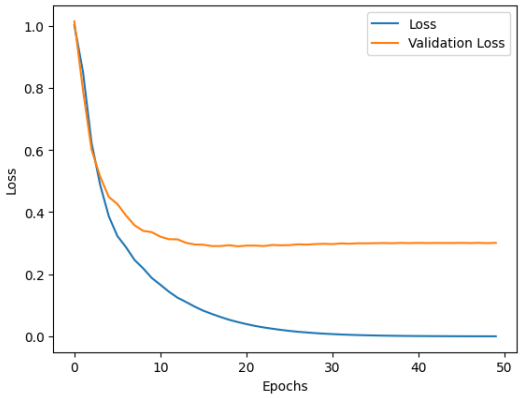

Figure 5.5: Training and Validation Loss of the Simple Autoencoder Model

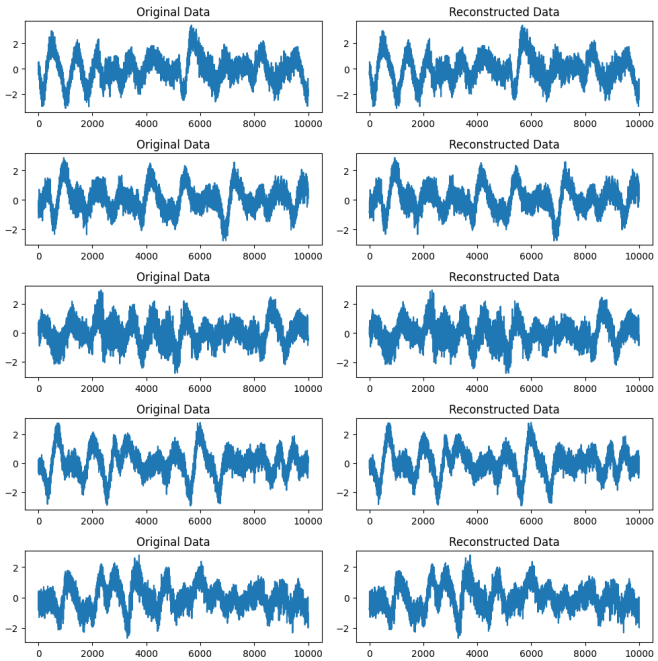

Figure 5.6: Training Results of the Simple Autoencoder Model

The simple autoencoder model yielded a reconstruction error of 0.166. Furthermore, R^2 explained variance, and correlation between the input data and reconstructed data equal 0.883, 0.883, and 0.940 respectively. Beyond that, random samples were chosen from the input data and reconstructed data in an effort to conduct a visual analysis if the pattern in the original data is preserved in the reconstructed data. Figure 5.6 indicates that the reconstructed data contains the pattern in the input data.

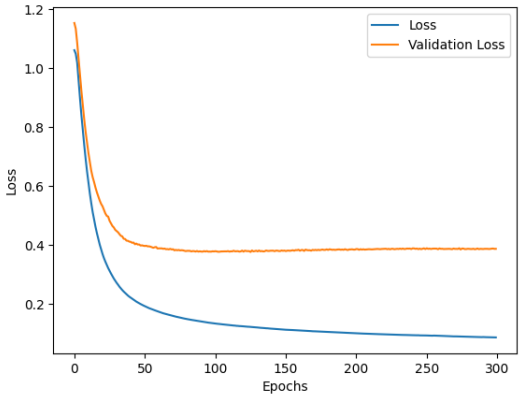

Figure 5.7: Training Results of the Sparse Autoencoder Model

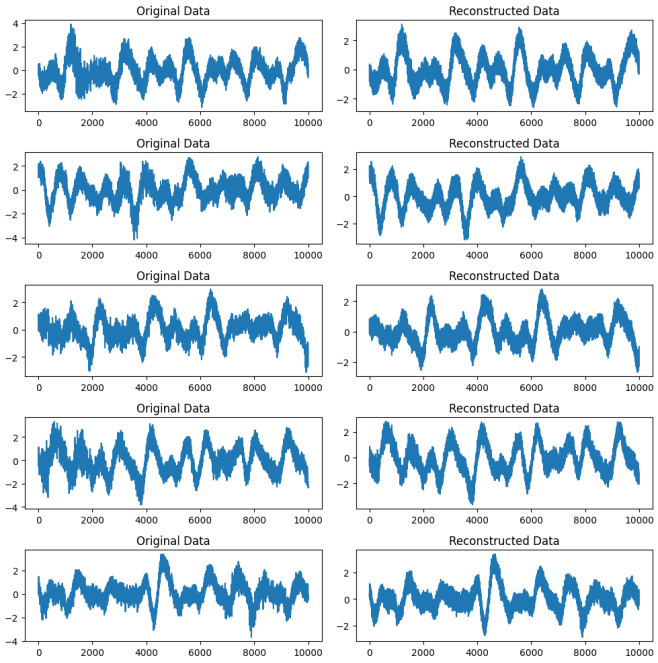

Figure 5.8: Training Results of the Sparse Autoencoder Model

According to Figure 5.7, training loss and validation loss decrease over time, and they stabilize after some epochs. The training time of the sparse autoencoder model corresponds to 21.095 seconds. The model yielded a reconstruction loss of 0.122. Beyond that, R^2 between the input data and the reconstructed data is equal to 0.877 while the explained variance is also equal to 0.877. Moreover, the correlation between the input data and the reconstructed data is 0.937.

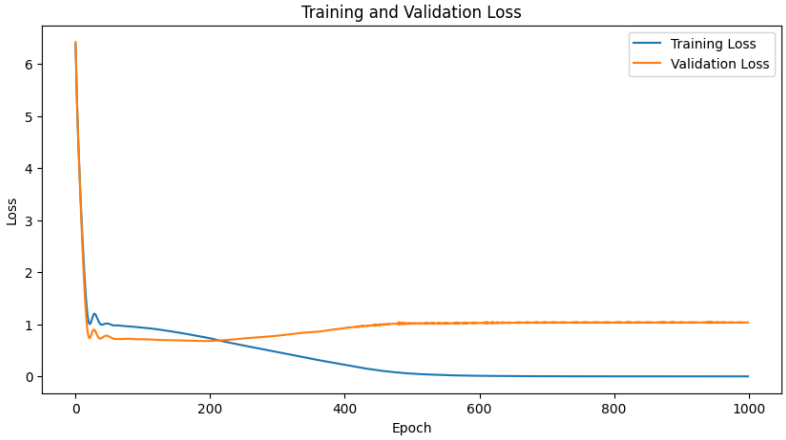

Figure 5.9: Training Results of the CNN Model Regarding Prediction of the Labels with Continuous Values

Figure 5.9 represents the training process of the CNN model in the context of the prediction of the labels with continuous values. The model was tested after training on the test set based on MSE and R^2. MSE equals 1.4092 while R^2 is equal to -0.92. The evaluation results indicate that the model can not predict the labels accurately.

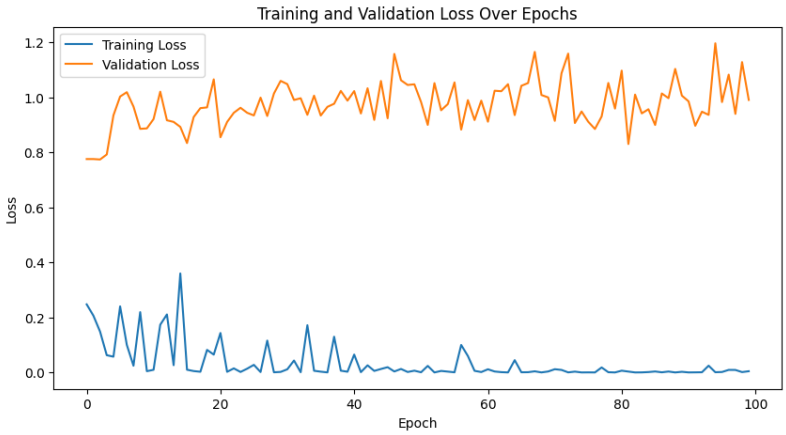

Figure 5.10: Training Results of the Deep Neural Network Model Regarding Prediction of the Labels with Continuous Values

Figure 5.10 shows that the validation loss does not converge. Moreover, an MSE score of 1.339 and R^2 of -0.437 highlight that the model can not capture the pattern in the data.

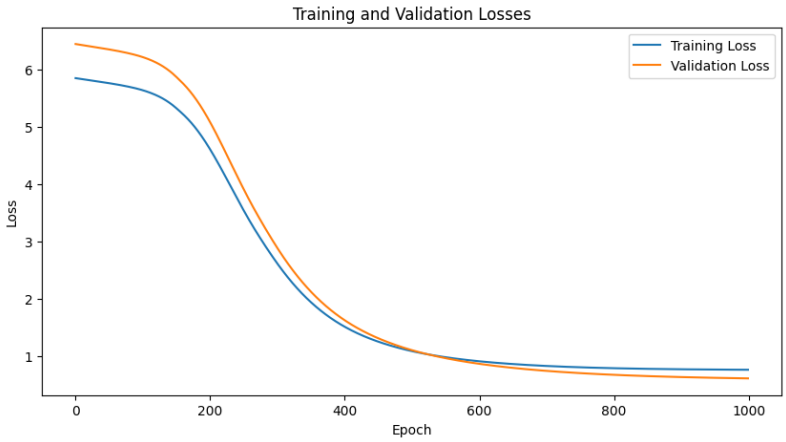

Figure 5.11: Training Results of the LSTM Model Regarding Prediction of the Labels with Continuous Values

Figure 5.11 depicts the training process of the LSTM model. After training the model was evaluated on the test set. After training the model yielded an MSE value of 0.75, which indicates that the model can not capture the pattern in the data accurately.

Henceforth, we will deal with the classification task. To transform the problem setting, into the classification of porous layers, we will set different threshold values to binarize continuous float labels. Specifically, we will conduct training sessions with three models across different threshold values, 1.0 and 2.0. Furthermore, we will resample the input data using random oversampling, and SMOTE to address the imbalanced data problem. By doing this, we aim to observe the effect of different threshold values on the model's performance. Additionally, we do not want to totally rely on random oversampling to tackle class imbalance, as in random oversampling samples are replicated. However, SMOTE generates synthetic samples. Therefore, we want to test the model's performance across two different resampling techniques.

To sum up, in the context of the regression task the models did not yield an adequate performance. By transforming the problem into a classification task we define two distinct classes, specifically porous, and non-porous.

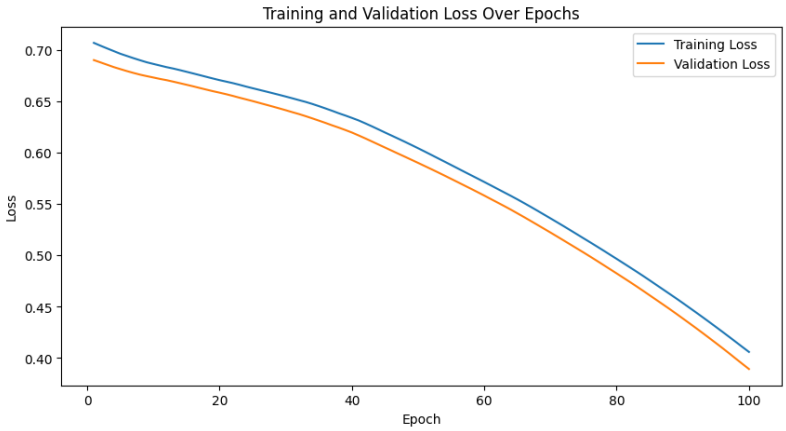

Figure 5.12: Performance Evaluation of Deep Neural Network Model for Binary Classification with a Threshold Value of 1.0 using Random Over Sampling

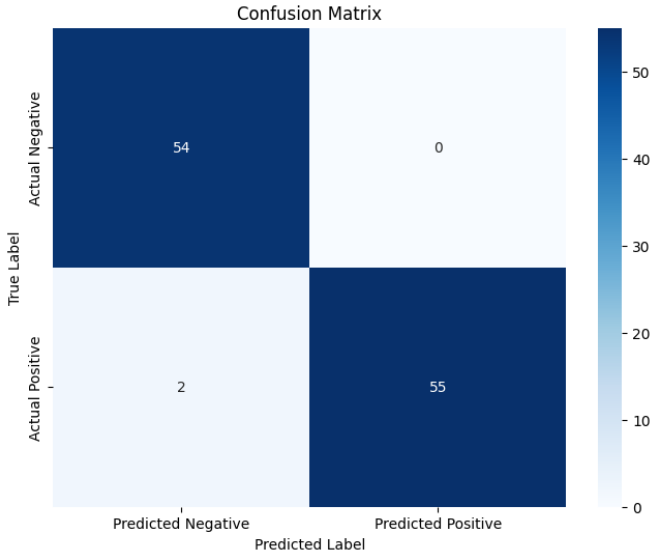

Figure 5.13: Confusion Matrix of Deep Neural Network Model for Binary Classification with a Threshold Value of 1.0 using Random Over Sampling

Figure 5.12 shows the training progress of the deep neural network model. The figure indicates that both training loss and validation loss decrease over time and converge to 0. The model produced an accuracy score of 0.9820 and an F1 score of 0.9821 on the test set. The confusion matrix regarding this classification task is depicted in Figure 5.13. The same model will be tested using a different resampling method to address the class imbalance, namely SMOTE. The study by Chawla et al. (2002) states that SMOTE generates synthetic samples instead of over-sampling with replacement. As SMOTE creates synthetic samples rather than replicating the existing samples, the model will be evaluated using SMOTE in addition to random over-sampling to test out the fitness of the models.

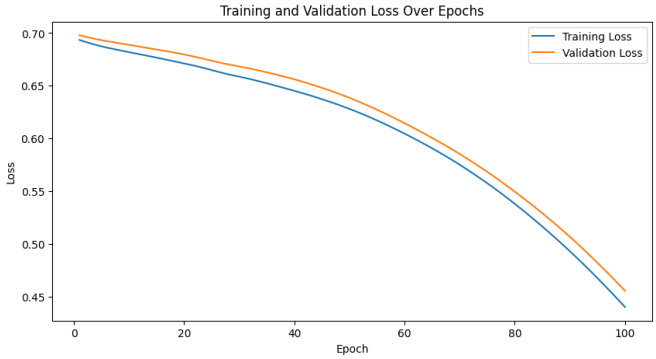

Figure 5.14: Performance Evaluation of Deep Neural Network Model for Binary Classification with a Threshold Value of 1.0 using SMOTE

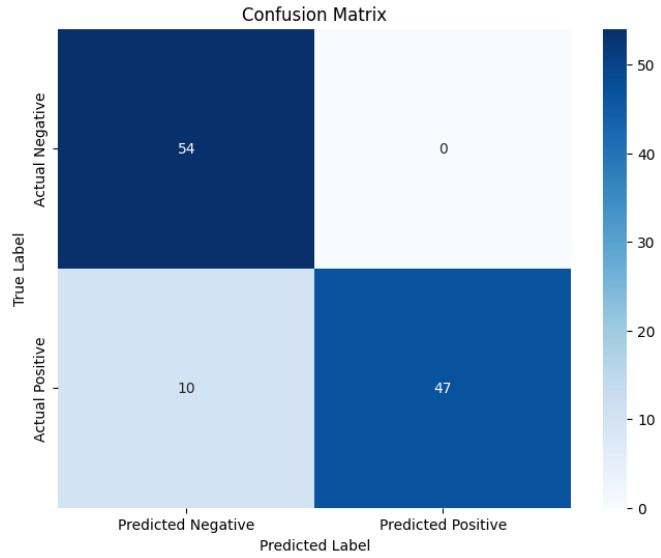

Figure 5.15: Confusion Matrix of Deep Neural Network Model for Binary Classification with a Threshold Value of 1.0 using SMOTE

Figure 5.14 indicates that training loss and validation loss shrink over time. After training the model was evaluated on the unseen test data, and it produced an accuracy of 0.9099 and an F1 score of 0.9038. This shows that the model can capture the pattern in the training data, and successfully distinguish the porous layers from the non-porous layers. Now, the deep learning model will be trained on the data with a threshold value of 2.0.

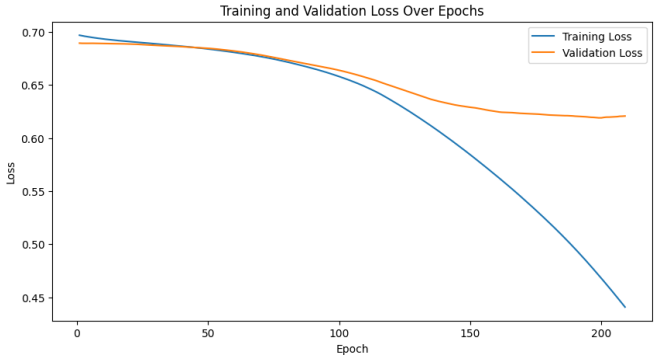

Figure 5.16: Performance Evaluation of Deep Neural Network Model for Binary Classification with a Threshold Value of 2.0 using Random Over Sampling

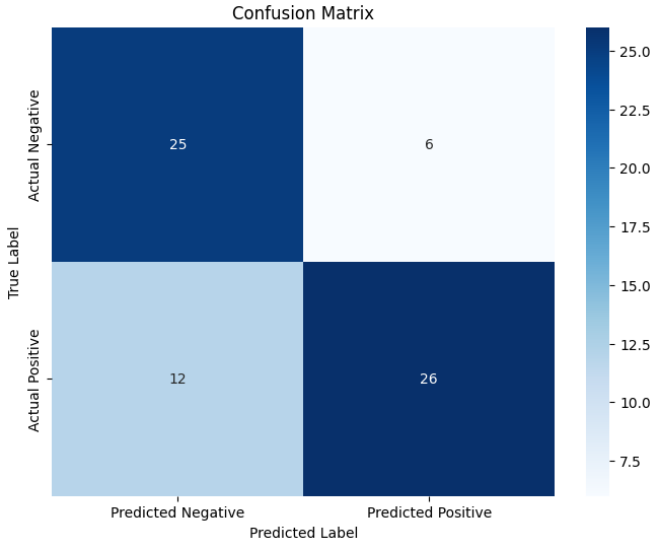

Figure 5.17: Confusion Matrix of Deep Neural Network Model for Binary Classification with a Threshold Value of 2.0 using Random Over Sampling

Figure 5.16 shows that the training loss decreases over epochs while the validation loss does not converge after 150th epoch. The model produced an accuracy score of 0.7391 and an F1-score of 0.7429 on the test set.

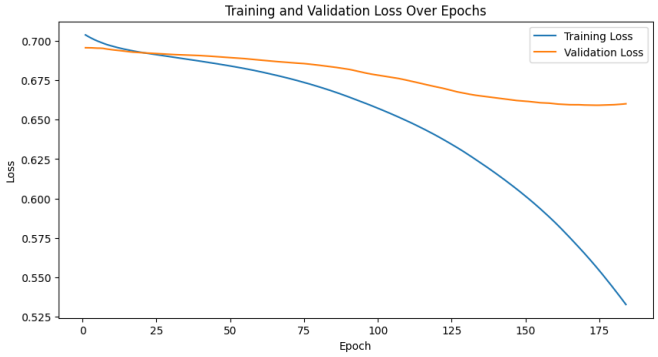

Figure 5.18: Performance Evaluation of Deep Neural Network Model for Binary Classification with a Threshold Value of 2.0 using SMOTE

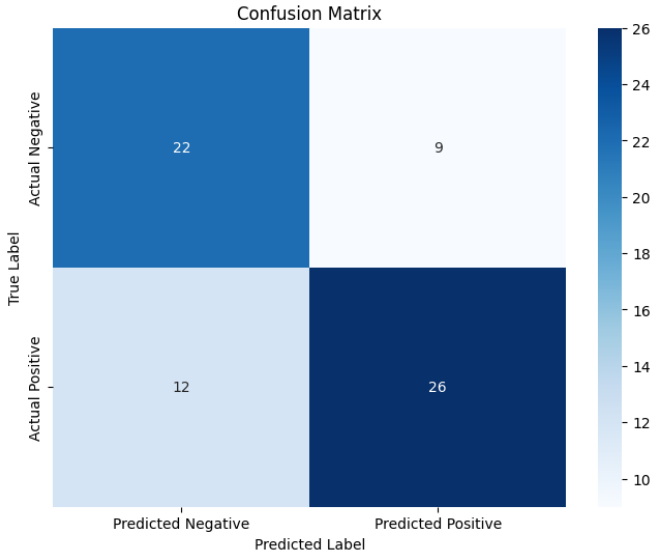

Figure 5.19: Confusion Matrix of Deep Neural Network Model for Binary Classification with a Threshold Value of 2.0 using SMOTE

When the deep learning model was trained regarding a threshold value of 2.0 and using SMOTE for resampling, it produced an accuracy score of 0.6957 and an F1-score of 0.7123 on the test set.

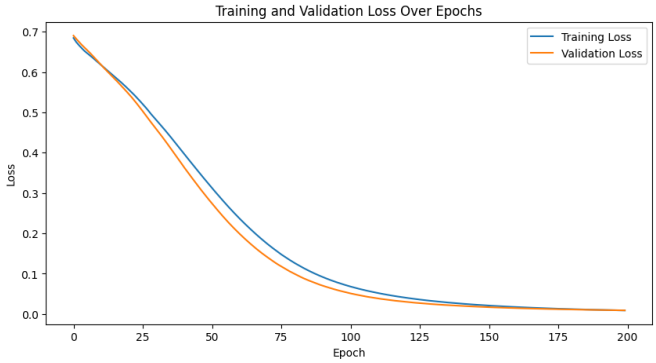

Figure 5.20: Performance Evaluation of the CNN model for Binary Classification with a Threshold Value of 1.0 using Random Over Samplig

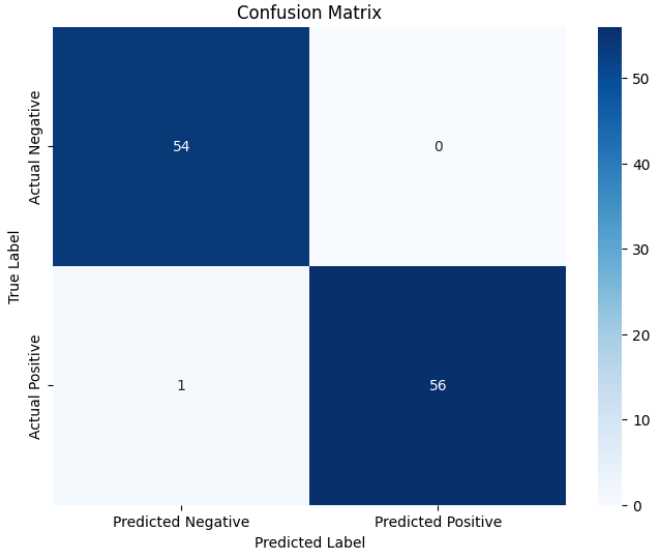

Figure 5.21: Confusion Matrix of the CNN model for Binary Classification with a Threshold Value of 1.0 using Random Over Samplig

According to Figure 5.20 both training loss and validation loss conver to 0 over some epochs. The CNN model produced an accuracy of 0.9910 and an F1-score of 0.9912 on the test set. These results indicate that the model can capture the pattern in the data and classify porous layers successfully. Now, we will train the model again using SMOTE to resample the data for the purpose of addressing the class imbalance problem.

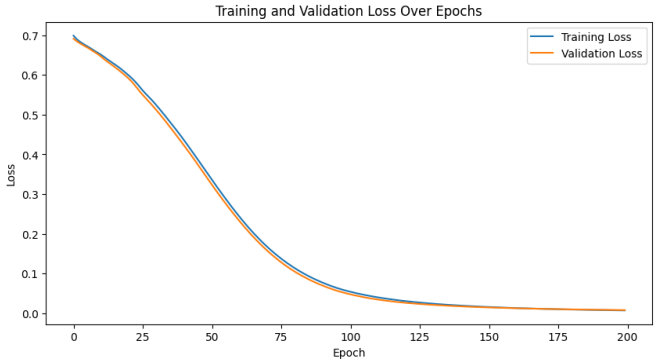

Figure 5.22: Performance Evaluation of the CNN model for Binary Classification with a Threshold Value of 1.0 using SMOTE

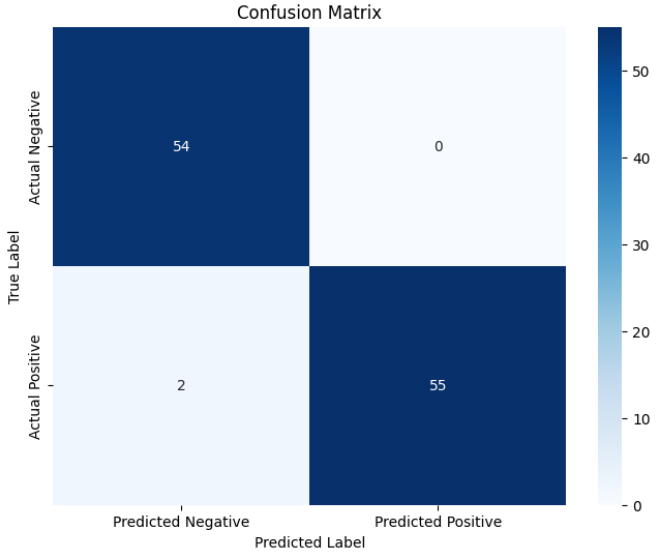

Figure 5.23: Confusion Matrix of the CNN model for Binary Classification with a Threshold Value of 1.0 using SMOTE

Figure 5.22 signifies the gradual decrease of training and validation losses over epochs. The CNN model produced an accuracy of 0.9820 and an F1-score of 0.9821 on the test set. These results denote that the CNN model can classify the labels correctly.

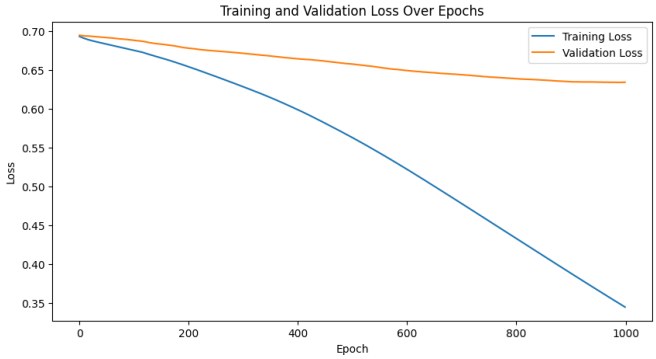

Figure 5.24: Performance Evaluation of the CNN model for Binary Classification with a Threshold Value of 2.0 using Random Over Sampling

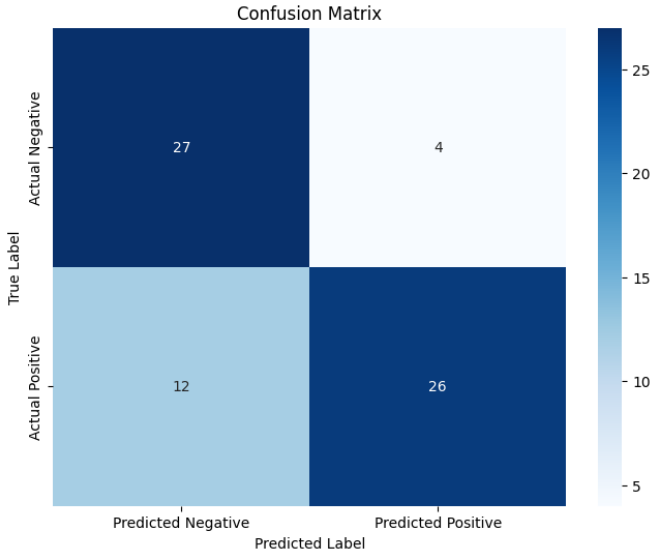

Figure 5.25: Confusion Matrix of the CNN model for Binary Classification with a Threshold Value of 2.0 using Random Over Sampling

According to Figure 5.24 training loss converges to 0 over epochs while validation loss does not decrease below 0.65. The model yielded an accuracy of 0.7681 and an F1-score of 0.7647 on the test set.

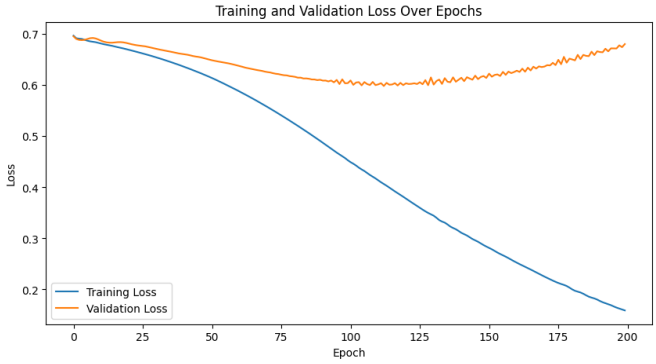

Figure 5.26: Performance Evaluation of the CNN model for Binary Classification with a Threshold Value of 2.0 using SMOTE

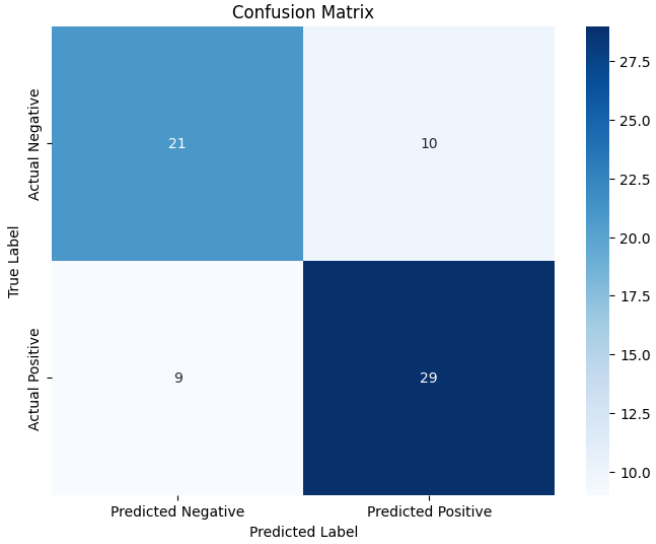

Figure 5.27: Confusion Matrix of the CNN model for Binary Classification with a Threshold Value of 2.0 using SMOTE

Regarding a threshold value of 2.0 and SMOTE for resampling the data, the model produced an accuracy of 0.7246 and an F1-score of 0.7532.

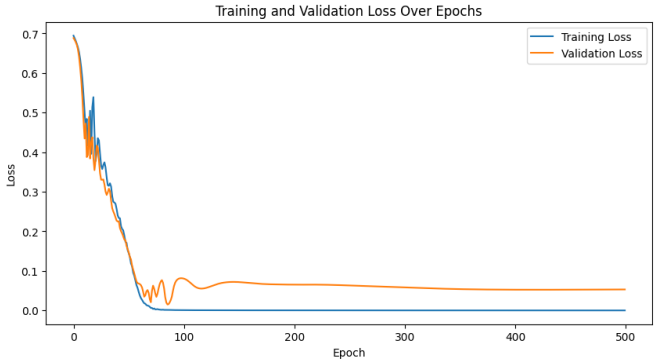

Figure 5.28: Performance Evaluation of the CNN-LSTM model for Binary Classification with a Threshold Value of 1.0 using Random Over Sampling

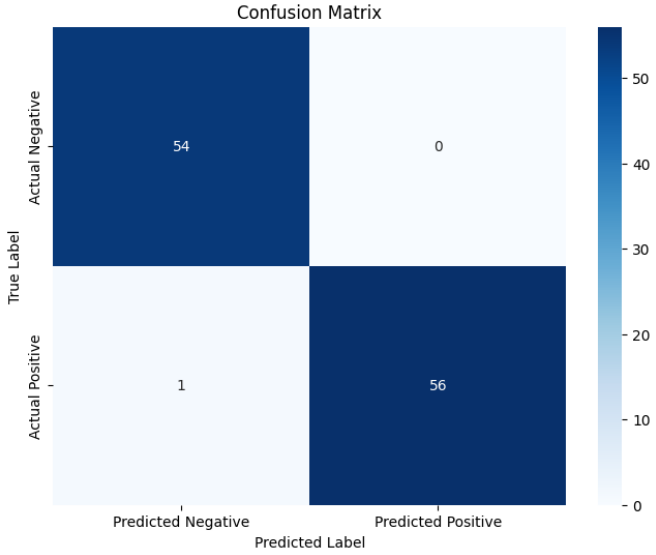

Figure 5.29: Confusion Matrix of the CNN-LSTM model for Binary Classification with a Threshold Value of 1.0 using Random Over Sampling

Figure 5.28 depicts training and validation loss of the CNN-LSTM model over 500 epochs. The figure shows that both training and validation loss decrease and converge to 0. The losses do not decrease further after 100^{th} epoch is passed. The model produced an accuracy of 0.9940, and an F1-score of 0.9943.

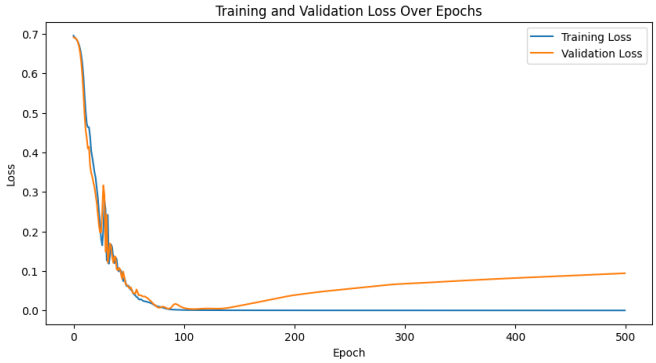

Figure 5.30: Performance Evaluation of the CNN-LSTM model for Binary Classification with a Threshold Value of 1.0 using SMOTE

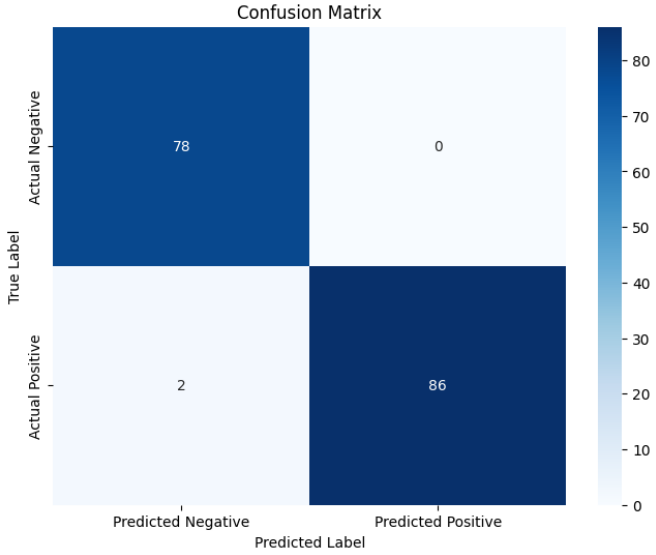

Figure 5.31: Confusion Matrix of the CNN-LSTM model for Binary Classification with a Threshold Value of 1.0 using SMOTE

When the model trained on the data that is resampled by using SMOTE for a threshold value of 1.0, the model produced an accuracy of 0.9880 and an F1-score of 0.9885.

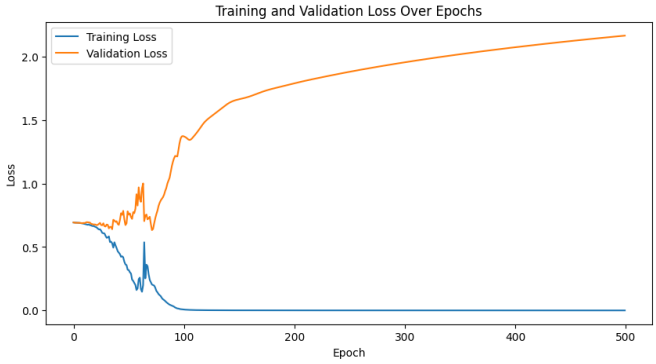

Figure 5.32: Performance Evaluation of the CNN-LSTM model for Binary Classification with a Threshold Value of 2.0 using Random Over Sampling

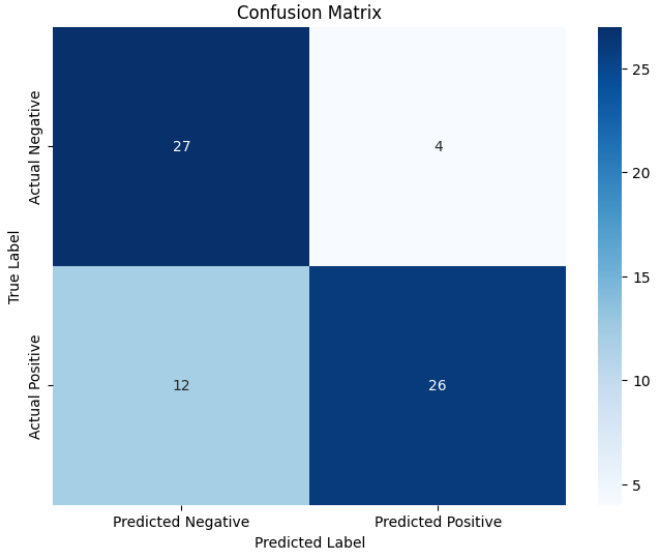

Figure 5.33: Confusion Matrix of the CNN-LSTM model for Binary Classification with a Threshold Value of 2.0 using Random Over Sampling

The CNN-LSTM model was tested out on the data for a threshold value of 2.0 using random over-sampling to tackle the imbalanced data problem. Figure 5.32 shows training and validation loss of the CNN-LSTM over 500 epochs. The figure indicates that the validation loss increases after 100^{th} epoch. The model returned an accuracy of 0.7087 and an F1-score of 0.7115.

Figure 5.34: Performance Evaluation of the CNN-LSTM model for Binary Classification with a Threshold Value of 2.0 using SMOTE

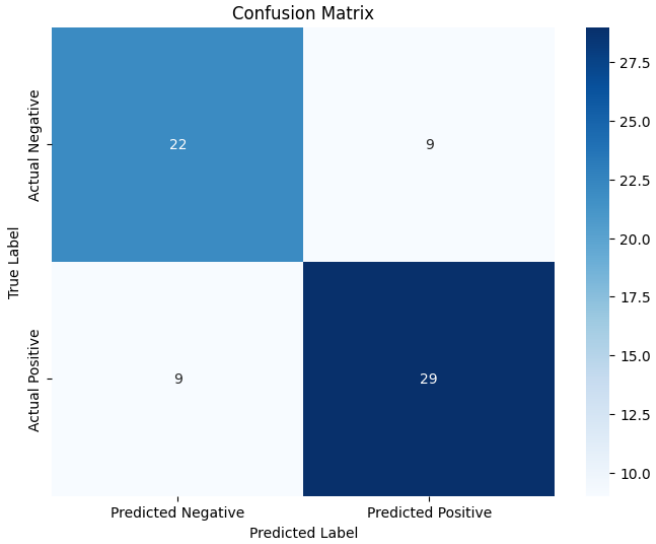

Figure 5.35: Confusion Matrix of the CNN-LSTM model for Binary Classification with a Threshold Value of 2.0 using SMOTE

The model achieved an accuracy of 0.7391 and an F1-score of 0.7632 on the test set when the input data was resampled using SMOTE for a threshold value of 2.0.

6 Discussion

In this study, various deep learning models were utilized for feature extraction, data compression, and for the purpose of predicting the pores. First, different autoencoder models, namely recurrent, convolutional, simple, and sparse models were utilized to compress the raw data. Then the encoded data which was generated by the autoencoder model that produced the lowest reconstruction loss, was used as input for predictive analysis.

Simple and sparse autoencoder models exhibited the best performance, while recurrent and convolutional autoencoder models failed to capture the pattern in the data. In general, simple and sparse autoencoders have a simple architecture and fewer parameters compared to recurrent and convolutional autoencoders. Furthermore, if the data represents a simple structure, models such as recurrent and convolutional autoencoders may enforce unnecessary complexity. In such cases, simpler models can demonstrate better performance in terms of reconstructing the input data. Moreover, the sparse autoencoder model applies regularization constraints on the encoded data, whereby emphasis on significant points regarding feature extraction is facilitated. Hence, this can reduce reconstruction loss. Additionally, if the time dependency and the sequential nature of the data do not demonstrate significance, the recurrent autoencoder model may have exhibited poor performance because of this.

After compressing the acoustic signals, we delved into two sorts of predictive analysis, in particular predicting continuous values of the labels and classifying porous layers. In predicting the continuous values of the labels none of the deep learning models yielded a satisfactory performance, while the models utilized for classifying porous and non-porous layers exhibited adequate performance in terms of accuracy and F1-score.

The difficulty of predicting continuous float labels by contrast to achieving good results in classification tasks can boil down to various reasons. Regression refers to predicting continuous values, which can be affected by the noise in the data. Thus, the nature of the regression problem can hinder the model from achieving high precision in predicting continuous values. Within the framework of this study, acoustic signals might contain noise. Consequently, this can demonstrate an obstacle regarding the prediction of continuous values.

Table 6.1: Model Performance in Classification

Model	Threshold	Resampling	Accuracy	F1-Score
Multi Layer Perceptron	1.0	Random Oversampling	0.9829	0.9821
	1.0	SMOTE	0.9099	0.9038
	2.0	Random Oversampling	0.7391	0.7429
	2.0	SMOTE	0.6957	0.7123
CNN	1.0	Random Oversampling	0.9910	0.9912
	1.0	SMOTE	0.9820	0.9821
	2.0	Random Oversampling	0.7681	0.7647
	2.0	SMOTE	0.7246	0.7532
CNN-LSTM	1.0	Random Oversampling	0.9940	0.9943
	1.0	SMOTE	0.9880	0.9885
	2.0	Random Oversampling	0.7087	0.7115
	2.0	SMOTE	0.7319	0.7632

Additionally, in case float labels are not boldly associated with the acoustic signals, it can make the prediction of continuous values an arduous task. Moreover, in classifying porous layers models classify data instances into two classes, which demonstrates an easier problem to deal with. Compared to regression, classification tasks might be less sensitive to noise in the input data, since we do not deal with prediction of continuous values. Furthermore, obtaining good results in regression tasks can require the implementation of more advanced feature engineering methods for capturing the important patterns and associations in the data.

To transform the problem into a classification task, threshold values were applied to the labels with continuous float values for binarization. Then, the performance of the deep learning models was tested based on different threshold values, specifically 1.0 and 2.0.

Table 6.1 depicts the performance of the three deep learning models across different threshold values and resampling techniques. All models demonstrated good performance when the threshold was set to 1.0. This means that the models can accurately classify data points. Beyond that, when the threshold value is increased to 2.0, a remarkable decrease in accuracy and F1-score is observed. What it may boil down to is that the nature of the data can influence the model's performance based on the decision boundary, namely the threshold value. It is likely that some patterns of the data are more noticeable when the threshold is equal to 1.0. This gives way to a better performance in terms of accuracy and F1 score. When the threshold is set to 2.0, the model may not be capable of capturing patterns and, hence can not relate the difference between class 0 and class 1.

To conclude, the decrease in the performance of the models might stem from the combined effect of choice of the threshold value, resampling methods, properties of the data, and class imbalance. However, the main reason for the drop in the performance of the classifiers seems to be the choice of the threshold value. The threshold value is interconnected with the decision boundary separating the porous and non-porous layers. All in all, we can conclude that the choice of an adequate threshold value is a crucial element in terms of achieving the targeted performance level.

7 Summary and Outlook

7.1 Summary

To sum up, this study delves into process monitoring in additive manufacturing using machine learning. Within the framework of this study, we tried to detect pores in the layers of the additive-manufactured part by directly predicting continuous float labels and transforming the problem into a classification task by thresholding the labels to categorize the data instances into class 0 and class 1.

First, raw acoustic signal data was processed by using autoencoder models for data compression and feature extraction purposes. Then, the encoded data by the autoencoder model was used as the input data to predict porous layers. Simple and sparse autoencoder models succeeded in reconstructing the data with low reconstruction loss. The regression task did not produce good results whereas deep learning models demonstrated good performance in the classification task. In the context of this study, we can conclude that for machine learning models it is easier to handle the classification task, as it revolves around categorizing data instances into predefined classes. To convert the problem of detecting porous layers into a classification task different threshold values were set, specifically 1.0 and 2.0. We observed that all models performed well when the threshold was set to 1.0, and the performance of the models dropped when the threshold was increased to 2.0. Thereby, we can conclude that the choice of the threshold value, and class imbalance have a significant influence on the model's performance.

Furthermore, this study makes a contribution to the field of automated detection of pores in additive manufacturing. By applying one of the machine learning models that demonstrated strong performance in the context of classifying pores, waste of materials can be reduced and deformities can be detected in the early stages of manufacturing. Early detection of the pores can reduce quality costs and waste of resources. Moreover, the overall quality of the additive-manufactured parts can be improved.

7.2 Outlook

This section focuses on suggestions for enhancing the performance of detecting pores in the context of additive manufacturing.

The performance of the constructed models can be improved by feeding them with data that is richer in quantity and diversity. This can be done by collecting data from various types of additive-manufactured products. Beyond that, different deep learning architectures that represent higher levels of sophistication and complexity can be explored and utilized in this field. Beyond that advanced feature engineering methods can be applied to the signal data to transform it. After data transformation, better results may be obtained by more precise modeling.

For instance, Liu et al. (2022) discusses ConNeXt which is a modernized version of ResNet. Liu et al. (2022) states that ConNeXt was developed based on ResNet across properties of transformers. According to the study by Liu et al. (2022) ConvNeXt can outperform vision and swin transformers in computer vision tasks. All in all, when the data is transformed by applying more sophisticated feature engineering techniques, and through the utilization of cutting-edge deep learning architectures like ConvNeXt, performance can be improved both in regression and classification tasks.

When the deep learning models are trained and validated on more diverse and larger datasets, they can be launched in real-production environments.

Bibliography

Abdulhameed, O., Al-Ahmari, A., Ameen, W. & Mian, S. H. (2019), „Additive manufacturing: Challenges, trends, and applications“, in: *Advances in Mechanical Engineering*, Bd. 11, Nr. 2, p. 1687814018822880. http://dx.doi.org/10.1177/1687814018822880.

Ahonen, T., Hadid, A. & Pietikainen, M. (2006), „Face description with local binary patterns: Application to face recognition“, in: *IEEE transactions on pattern analysis and machine intelligence*, Bd. 28, Nr. 12, p. 2037–2041.

Alpaydin, E. (2020), *Introduction to machine learning*, MIT press.

Asuero, A. G., Sayago, A. & González, A. G. (2006), „The Correlation Coefficient: An Overview“, in: *Critical Reviews in Analytical Chemistry*, Bd. 36, Nr. 1, p. 41–59. http://dx.doi.org/10.1080/10408340500526766.

Bai, X., Wang, X., Liu, X., Liu, Q., Song, J., Sebe, N. & Kim, B. (2021), „Explainable deep learning for efficient and robust pattern recognition: A survey of recent developments“, in: *Pattern Recognition*, Bd. 120, p. 108102. http://dx.doi.org/https://doi.org/10.1016/j.patcog.2021.108102.

Bank, D., Koenigstein, N. & Giryes, R. (2023), „Autoencoders“, in: *Machine Learning for Data Science Handbook: Data Mining and Knowledge Discovery Handbook*, p. 353–374.

Bartholomew, D. J., Knott, M. & Moustaki, I. (2011), *Latent variable models and factor analysis: A unified approach*, Bd. 904, John Wiley & Sons.

Bengio, Y., Simard, P. & Frasconi, P. (1994), „Learning long-term dependencies with gradient descent is difficult“, in: *IEEE Transactions on Neural Networks*, Bd. 5, Nr. 2, p. 157–166. http://dx.doi.org/10.1109/72.279181.

Bi, Q., Goodman, K. E., Kaminsky, J. & Lessler, J. (2019), „What is Machine Learning? A Primer for the Epidemiologist“, in: *American Journal of Epidemiology*, Bd. 188, Nr. 12, p. 2222–2239. http://dx.doi.org/10.1093/aje/kwz189.

Bishop, C. M. & Nasrabadi, N. M. (2006), *Pattern recognition and machine learning*, Bd. 4, Springer.

Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt, B. & Varoquaux, G. (2013), „API design for machine learning software: experiences from the scikit-learn project“, in: *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*p. 108–122.

Caggiano, A., Zhang, J., Alfieri, V., Caiazzo, F., Gao, R. & Teti, R. (2019), „Machine learning-based image processing for on-line defect recognition in additive manufacturing“, in: *CIRP annals*, Bd. 68, Nr. 1, p. 451–454.

Chandola, V., Banerjee, A. & Kumar, V. (2009), „Anomaly detection: A survey“, in: *ACM computing surveys (CSUR)*, Bd. 41, Nr. 3, p. 1–58.

Chawla, N. V., Bowyer, K. W., Hall, L. O. & Kegelmeyer, W. P. (2002), „SMOTE: synthetic minority over-sampling technique“, in: *Journal of artificial intelligence research*, Bd. 16, p. 321–357.

Cheepu, M. (2023), „Machine learning approach for the prediction of defect characteristics in wire arc additive manufacturing“, in: *Transactions of the Indian Institute of Metals*, Bd. 76, Nr. 2, p. 447–455.

Chen, K., Zhou, Y. & Dai, F. (2015), „A LSTM-based method for stock returns prediction: A case study of China stock market“, in: *2015 IEEE international conference on big data (big data)*IEEE, p. 2823–2824.

Chung, J., Gulcehre, C., Cho, K. & Bengio, Y. (2014), „Empirical evaluation of gated recurrent neural networks on sequence modeling“, in: *arXiv preprint arXiv:1412.3555*.

Cooper, K. (2001), „Laser Engineered Net Shaping“, in: *Rapid prototyping technology*

selection and application New York: Marcel Dekker.

de Leon, A. C., Chen, Q., Palaganas, N. B., Palaganas, J. O., Manapat, J. & Advincula, R. C. (2016), „High performance polymer nanocomposites for additive manufacturing applications“, in: *Reactive and Functional Polymers*, Bd. 103, p. 141–155. http://dx.doi.org/https://doi.org/10.1016/j.reactfunctpolym.2016.04.010.

DebRoy, T., Wei, H., Zuback, J., Mukherjee, T., Elmer, J., Milewski, J., Beese, A. M., Wilson-Heid, A. d., De, A. & Zhang, W. (2018), „Additive manufacturing of metallic components–process, structure and properties“, in: *Progress in Materials Science*, Bd. 92, p. 112–224.

Diaz-Papkovich, A., Anderson-Trocmé, L. & Gravel, S. (2021), „A review of UMAP in population genetics“, in: *Journal of Human Genetics*, Bd. 66, Nr. 1, p. 85–91.

Dilberoglu, U. M., Gharehpapagh, B., Yaman, U. & Dolen, M. (2017), „The role of additive manufacturing in the era of industry 4.0“, in: *Procedia manufacturing*, Bd. 11, p. 545–554.

Duda, R. O., Hart, P. E. et al. (1973), *Pattern classification and scene analysis*, Bd. 3, Wiley New York.

Eschner, N., Weiser, L., Häfner, B. & Lanza, G. (2019), „Akustische Prozessüberwachung für das Laserstrahlschmelzen (LBM) mit neuronalen Netzen: Eine Potentialbewertung“, in: *tm-Technisches Messen*, Bd. 86, Nr. 11, p. 661–672.

Everton, S. K., Hirsch, M., Stravroulakis, P., Leach, R. K. & Clare, A. T. (2016), „Review of in-situ process monitoring and in-situ metrology for metal additive manufacturing“, in: *Materials & Design*, Bd. 95, p. 431–445.

Folk, M., Heber, G., Koziol, Q., Pourmal, E. & Robinson, D. (2011), „An overview of the HDF5 technology suite and its applications“, in: *Proceedings of the EDBT/ICDT 2011 workshop on array databases*p. 36–47.

Gers, F. A., Schmidhuber, J. & Cummins, F. (2000), „Learning to forget: Continual prediction with LSTM“, in: *Neural computation*, Bd. 12, Nr. 10, p. 2451–2471.

Godec, D., Gonzalez-Gutierrez, J., Nordin, A., Pei, E. & Ureña Alcázar, J. (2022), *A Guide to*

Additive Manufacturing, Springer Nature.

Goodfellow, I., Bengio, Y. & Courville, A. (2016), *Deep learning*, MIT press.

Grasso, M. & Colosimo, B. M. (2017), „Process defects and in situ monitoring methods in metal powder bed fusion: a review“, in: *Measurement Science and Technology*, Bd. 28, Nr. 4, p. 044005.

Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., Wang, G., Cai, J. & Chen, T. (2018), „Recent advances in convolutional neural networks“, in: *Pattern Recognition*, Bd. 77, p. 354–377. http://dx.doi.org/https://doi.org/10.1016/j.patcog.2017.10.013.

Guo, X., Yin, Y., Dong, C., Yang, G. & Zhou, G. (2008), „On the class imbalance problem“, in: *2008 Fourth international conference on natural computation*Bd. 4, IEEE, p. 192–201.

Harris, C. R., Millman, K. J., Van Der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J. et al. (2020), „Array programming with NumPy“, in: *Nature*, Bd. 585, Nr. 7825, p. 357–362.

Hennig, C., Meila, M., Murtagh, F. & Rocci, R. (2015), *Handbook of cluster analysis*, CRC press.

Hernavs, J., Ficko, M., Berus, L., Rudolf, R. & Klančnik, S. (2018), „Deep Learning in Industry 4.0—Brief Overview“, in: *J. Prod. Eng*, Bd. 21, Nr. 2, p. 1–5.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. R. (2012), „Improving neural networks by preventing co-adaptation of feature detectors“, in: *arXiv preprint arXiv:1207.0580*.

Hochreiter, S. & Schmidhuber, J. (1997), „Long short-term memory“, in: *Neural computation*, Bd. 9, Nr. 8, p. 1735–1780.

Huang, C., Wang, J., Wang, S. & Zhang, Y. (2023), „A review of deep learning in dentistry“, in: *Neurocomputing*, Bd. 554, p. 126629. http://dx.doi.org/https://doi.org/10.1016/j.neucom.2023.126629.

Huang, S. H., Liu, P., Mokasdar, A. & Hou, L. (2013), „Additive manufacturing and its societal impact: a literature review“, in: *The International journal of advanced manufacturing technology*, Bd. 67, p. 1191–1203.

Janiesch, C., Zschech, P. & Heinrich, K. (2021), „Machine learning and deep learning“, in: *Electronic Markets*, Bd. 31, Nr. 3, p. 685–695.

Johnson, J. M. & Khoshgoftaar, T. M. (2019), „Survey on deep learning with class imbalance“, in: *Journal of Big Data*, Bd. 6, Nr. 1, p. 1–54.

Jordan, M. I. & Mitchell, T. M. (2015), „Machine learning: Trends, perspectives, and prospects“, in: *Science*, Bd. 349, Nr. 6245, p. 255–260. http://dx.doi.org/10.1126/science.aaa8415.

Khanzadeh, M., Chowdhury, S., Marufuzzaman, M., Tschopp, M. A. & Bian, L. (2018), „Porosity prediction: Supervised-learning of thermal history for direct laser deposition“, in: *Journal of Manufacturing Systems*, Bd. 47, p. 69–82. http://dx.doi.org/https://doi.org/10.1016/j.jmsy.2018.04.001.

Kochan, A. (1997), „Features Rapid growth for rapid prototyping“, in: *Assembly Automation*, Bd. 17, Nr. 3, p. 215–217.

Kothari, J. D. (2018), „Detecting welding defects in steel plates using machine learning and computer vision algorithms“, in: *Jubin Dipakkumar Kothari (2018). Detecting Welding Defects in Steel Plates using Machine Learning and Computer Vision Algorithms. International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, Bd. 7, Nr. 9, p. 3682–3686.

Kuester, J., Gross, W. & Middelmann, W. (2021), „1D-convolutional autoencoder based hyperspectral data compression“, in: *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Bd. 43, p. 15–21.

Kühl, N., Hirt, R., Baier, L., Schmitz, B. & Satzger, G. (2021), „How to conduct rigorous supervised machine learning in information systems research: the supervised machine learning report card“, in: *Communications of the Association for Information Systems*, Bd. 48, Nr. 1, p. 46.

Kumar, V. & Dutta, D. (1997), „An assessment of data formats for layered manufacturing“, in: *Advances in Engineering Software*, Bd. 28, Nr. 3, p. 151–164.

Lasi, H., Fettke, P., Kemper, H.-G., Feld, T. & Hoffmann, M. (2014), „Industry 4.0“, in: *Business & information systems engineering*, Bd. 6, p. 239–242.

LeCun, Y., Bengio, Y. & Hinton, G. (2015), „Deep learning“, in: *nature*, Bd. 521, Nr. 7553, p. 436–444.

LeCun, Y., Bottou, L., Orr, G. & Müller, K. (2012), „Efficient backprop In: Neural Networks: Tricks of the Trade, 9–48“, in: *Springer*, Bd. 10, p. 3–540.

Lemaître, G., Nogueira, F. & Aridas, C. K. (2017), „Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning“, in: *The Journal of Machine Learning Research*, Bd. 18, Nr. 1, p. 559–563.

Li, Z., Liu, F., Yang, W., Peng, S. & Zhou, J. (2021), „A survey of convolutional neural networks: analysis, applications, and prospects“, in: *IEEE transactions on neural networks and learning systems*.

Lindeberg, T. (2012), „Scale invariant feature transform“, in: .

Lipton, Z. C., Berkowitz, J. & Elkan, C. (2015), „A critical review of recurrent neural networks for sequence learning“, in: *arXiv preprint arXiv:1506.00019*.

Liu, Z., Mao, H., Wu, C.-Y., Feichtenhofer, C., Darrell, T. & Xie, S. (2022), „A convnet for the 2020s“, in: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*p. 11976–11986.

McInnes, L., Healy, J. & Melville, J. (2018), „Umap: Uniform manifold approximation and projection for dimension reduction“, in: *arXiv preprint arXiv:1802.03426*.

Michelucci, U. (2022), „An introduction to autoencoders“, in: *arXiv preprint arXiv:2201.03898*.

Nair, V. & Hinton, G. E. (2010), „Rectified linear units improve restricted boltzmann machines“, in: *Proceedings of the 27th international conference on machine learning (ICML-10)*p. 807–

814.

Nalajam, P. K. & V, R. (2021), „Microstructural porosity segmentation using machine learning techniques in wire-based direct energy deposition of AA6061“, in: *Micron*, Bd. 151, p. 103161. http://dx.doi.org/https://doi.org/10.1016/j.micron.2021.103161.

Noorani, R. (2006), „Rapid prototyping: principles and applications“, in: *(No Title)*.

Oliphant, T. E. et al. (2006), *Guide to numpy*, Bd. 1, Trelgol Publishing USA.

Pascanu, R., Mikolov, T. & Bengio, Y. (2013), „On the difficulty of training recurrent neural networks“, in: *International conference on machine learning*Pmlr, p. 1310–1318.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J. & Chintala, S. (2019a), „PyTorch: An Imperative Style, High-Performance Deep Learning Library“, in: *Advances in Neural Information Processing Systems 32*, Hrsg. H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox & R. Garnett, Curran Associates, Inc., p. 8024–8035. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L. et al. (2019b), „Pytorch: An imperative style, high-performance deep learning library“, in: *Advances in neural information processing systems*, Bd. 32.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. & Duchesnay, E. (2011), „Scikit-learn: Machine Learning in Python“, in: *Journal of Machine Learning Research*, Bd. 12, p. 2825–2830.

Picek, S., Heuser, A., Jovic, A., Bhasin, S. & Regazzoni, F. (2019), „The curse of class imbalance and conflicting metrics with machine learning for side-channel evaluations“, in: *IACR Transactions on Cryptographic Hardware and Embedded Systems*, p. 209–237.

Prakash, K. S., Nancharaih, T. & Rao, V. S. (2018), „Additive Manufacturing Techniques in Manufacturing -An Overview“, in: *Materials Today: Proceedings*, Bd. 5, Nr. 2, Part 1, p.

3873–3882, 7th International Conference of Materials Processing and Characterization, March 17-19, 2017. http://dx.doi.org/https://doi.org/10.1016/j.matpr.2017.11.642.

Prati, R. C., Batista, G. E. & Monard, M. C. (2009), „Data mining with imbalanced class distributions: concepts and methods.“, in: *IICAI*p. 359–376.

Proteau, A., Zemouri, R., Tahan, A. & Thomas, M. (2020), „Dimension reduction and 2D-visualization for early change of state detection in a machining process with a variational autoencoder approach“, in: *The International Journal of Advanced Manufacturing Technology*, Bd. 111, p. 3597–3611.

Qin, J., Hu, F., Liu, Y., Witherell, P., Wang, C. C., Rosen, D. W., Simpson, T. W., Lu, Y. & Tang, Q. (2022), „Research and application of machine learning for additive manufacturing“, in: *Additive Manufacturing*, Bd. 52, p. 102691. http://dx.doi.org/https://doi.org/10.1016/j.addma.2022.102691.

Ren, S., He, K., Girshick, R. & Sun, J. (2017), „Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks“, in: *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Bd. 39, Nr. 6, p. 1137–1149. http://dx.doi.org/10.1109/TPAMI.2016.2577031.

Ren, W., Wen, G., Zhang, Z. & Mazumder, J. (2022), „Quality monitoring in additive manufacturing using emission spectroscopy and unsupervised deep learning“, in: *Materials and Manufacturing Processes*, Bd. 37, Nr. 11, p. 1339–1346.

Rokach, L., Maimon, O. & Shmueli, E. (2023), „Machine Learning for Data Science Handbook“, in: .

Rumelhart, D., Hinton, G. & Williams, R. (1986), „Learning internal representations by error propagation. Parallel Distributed Processing (PDP): Exploration in the Microstructure of Cognition", Vol. 1, Chapter 8“, in: *MIT Press, Cambridge, Massachusetts*, Bd. 98, p. 318–362.

Sakurada, M. & Yairi, T. (2014), „Anomaly detection using autoencoders with nonlinear dimensionality reduction“, in: *Proceedings of the MLSDA 2014 2nd workshop on machine learning for sensory data analysis*p. 4–11.

Samuel, A. L. (1959), „Some studies in machine learning using the game of checkers“, in: *IBM Journal of research and development*, Bd. 3, Nr. 3, p. 210–229.

Satterlee, N., Torresani, E., Olevsky, E. & Kang, J. S. (2022), „Comparison of machine learning methods for automatic classification of porosities in powder-based additive manufactured metal parts“, in: *The International Journal of Advanced Manufacturing Technology*, Bd. 120, p. 6761–6776.

Satterlee, N., Torresani, E., Olevsky, E. & Kang, J. S. (2023), „Automatic detection and characterization of porosities in cross-section images of metal parts produced by binder jetting using machine learning and image augmentation“, in: *Journal of Intelligent Manufacturing*, p. 1–23.

Shamsaei, N., Yadollahi, A., Bian, L. & Thompson, S. M. (2015), „An overview of Direct Laser Deposition for additive manufacturing; Part II: Mechanical behavior, process parameter optimization and control“, in: *Additive manufacturing*, Bd. 8, p. 12–35.

Singh, A. & Jain, A. (2019), „An Empirical Study of AML Approach for Credit Card Fraud Detection—Financial Transactions“, in: *International Journal of Computers Communications & Control*, Bd. 14, Nr. 6, p. 670–690.

Smith, L. N. (2018), „A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay“, in: *arXiv preprint arXiv:1803.09820*.

Smoqi, Z., Gaikwad, A., Bevans, B., Kobir, M. H., Craig, J., Abul-Haj, A., Peralta, A. & Rao, P. (2022), „Monitoring and prediction of porosity in laser powder bed fusion using physics-informed meltpool signatures and machine learning“, in: *Journal of Materials Processing Technology*, Bd. 304, p. 117550. http://dx.doi.org/https://doi.org/10.1016/j.jmatprotec.2022.117550.

Sutskever, I., Martens, J. & Hinton, G. E. (2011), „Generating text with recurrent neural networks“, in: *Proceedings of the 28th international conference on machine learning (ICML-11)*p. 1017–1024.

Sutton, R. S. & Barto, A. G. (2018), *Reinforcement learning: An introduction*, MIT press.

Tapia, G. & Elwany, A. (2014), „A Review on Process Monitoring and Control in Metal-Based Additive Manufacturing“, in: *Journal of Manufacturing Science and Engineering*, Bd. 136, Nr. 6, p. 060801. http://dx.doi.org/10.1115/1.4028540.

The pandas development team (), *pandas-dev/pandas: Pandas*. https://github.com/pandas-dev/pandas.

Thompson, M. K., Moroni, G., Vaneker, T., Fadel, G., Campbell, R. I., Gibson, I., Bernard, A., Schulz, J., Graf, P., Ahuja, B. et al. (2016), „Design for Additive Manufacturing: Trends, opportunities, considerations, and constraints“, in: *CIRP annals*, Bd. 65, Nr. 2, p. 737–760.

Thomsen, E. (2002), *OLAP solutions: building multidimensional information systems*, John Wiley & Sons.

Wang, P., Liu, H., Wang, L. & Gao, R. X. (2018), „Deep learning-based human motion recognition for predictive context-aware human-robot collaboration“, in: *CIRP Annals*, Bd. 67, Nr. 1, p. 17–20. http://dx.doi.org/https://doi.org/10.1016/j.cirp.2018.04.066.

Weimer, D., Scholz-Reiter, B. & Shpitalni, M. (2016), „Design of deep convolutional neural network architectures for automated feature extraction in industrial inspection“, in: *CIRP Annals*, Bd. 65, Nr. 1, p. 417–420. http://dx.doi.org/https://doi.org/10.1016/j.cirp.2016.04.072.

Wu, D., Wei, Y. & Terpenny, J. (2018), „Surface roughness prediction in additive manufacturing using machine learning“, in: *International Manufacturing Science and Engineering Conference*Bd. 51371, American Society of Mechanical Engineers, p. V003T02A018.

Yang, J., Chen, Y., Huang, W. & Li, Y. (2017), „Survey on artificial intelligence for additive manufacturing“, in: *2017 23rd international conference on automation and computing (ICAC)*IEEE, p. 1–6.

Zawadzki, P. & Żywicki, K. (2016), „Smart product design and production control for effective mass customization in the Industry 4.0 concept“, in: *Management and production engineering review*.

Zhang, B., Liu, S. & Shin, Y. (2019), *In-process monitoring of porosity during laser additive manufacturing process. Addit Manuf 28: 497–505*.

Zhang, H., Zhang, L. & Jiang, Y. (2019), „Overfitting and underfitting analysis for deep learning based end-to-end communication systems“, in: *2019 11th international conference on wireless communications and signal processing (WCSP)*IEEE, p. 1–6.

Zhang, Y. (2018), „A better autoencoder for image: Convolutional autoencoder“, in: *ICONIP17-DCEC. Available online: http://users. cecs. anu. edu. au/Tom. Gedeon/conf/ABCs2018/paper/ABCs2018_paper_58. pdf (accessed on 23 March 2017)*.

Zhao, Z., Chen, W., Wu, X., Chen, P. C. & Liu, J. (2017), „LSTM network: a deep learning approach for short-term traffic forecast“, in: *IET Intelligent Transport Systems*, Bd. 11, Nr. 2, p. 68–75.

Zhu, X. & Goldberg, A. B. (2022), *Introduction to semi-supervised learning*, Springer Nature.

List of Figures

List of Tables