CHALMERS
UNIVERSITY OF TECHNOLOGY

Autoencoders for Anomaly Detection in Commercial Truck Gearshifts

Applying Gearshift Classification and Anomaly Detection in Commercial Trucks Using Autoencoder Neural Networks

Master's thesis in Complex Adaptive Systems

ANES IMSIROVIC

ARVIN ROKNI

DEPARTMENT OF ELECTRICAL ENGINEERING

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2024
www.chalmers.se

Master's thesis 2024

Autoencoders for Anomaly Detection in Commercial Truck Gearshifts

Applying Gearshift Classification and Anomaly Detection in Commercial Trucks Using Autoencoder Neural Networks

ANES IMSIROVIC

ARVIN ROKNI

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of Communication, Antennas and Optical Networks
Chalmers University of Technology
Gothenburg, Sweden 2024

Autoencoders for Anomaly Detection in Commercial Truck Gearshifts
Applying Gearshift Classification and Anomaly Detection in Commercial Trucks Using Autoencoder Neural Networks
ANES IMSIROVIC
ARVIN ROKNI

Supervisor: Oscar Olesen , Volvo GTT
Supervisor: Simon Lillskog , Volvo GTT
Supervisor: Alireza Bordbar , Department of Electrical Engineering
Examiner: Fredrik Brännström, Department of Electrical Engineering

Master's Thesis 2024
Department of Electrical Engineering
Division of Communication, Antennas and Optical Networks
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2024

Autoencoders for Anomaly Detection in Commercial Truck Gearshifts
Applying Gearshift Classification and Anomaly Detection in Commercial Trucks Using Autoencoder Neural Networks
ANES IMSIROVIC
ARVIN ROKNI
Department of Electrical Engineering
Chalmers University of Technology

Abstract

The following study revolves around implementing a machine learning model that supports engineers in analyzing vehicle sensor data at Volvo Trucks. The aim is to classify different gear shift types and to automate the detection of abnormal gear shifts making decisions more efficient and data-driven. By applying clustering- and classification algorithms, the model demonstrates how unlabeled field test data can be classified in diverse gear shift types. Furthermore, through evaluation and comparision of dense, convolutional and long short-term memory (LSTM) autoencoder (AE) neural networks, this study exhibits how abnormal sensor data within specific gear shift types can be detected. Obtained results indicate adequate gear shift classification with an overall accuracy of 98%. Furthermore, a comparison of the autoencoders in regard to the performance metrics accuracy, precision and recall concluded that the convolutional autoencoder outperformed the other architectures with scores above 95%. Although further fine-tuning of the implemented model is possible, findings indicate that it is feasible to develop and use machine learning models for classification and anomaly detection of gear shift data within heavy-duty trucks.

Keywords: Machine Learning, Deep Neural Networks, Autoencoders, Sensor Data, Unsupervised, Clustering, Classification, Anomaly Detection, Truck Gear Shifts.

Acknowledgements

Before going any further, we would like to acknowledge the people who have given their support to this work. We would like to start by thanking all the staff at Volvo Trucks who took part in this work and emphasize the excellent work environment that we have been offered during the course of the work. Special thanks go to Oscar Olesen, Simon Lillskog and Johan Jonsson, whose presence this work would not have been possible without. We are grateful for all the valuable meetings and useful feedback that has been provided to us.

Furthermore, we would like to emphasize the guidance we received from our supervisor Alireza Bordbar at Chalmers. We are grateful for all the time and energy he has put into suggesting improvements and ensuring that the work maintains the academic level demanded by the university.

Anes Imsirovic & Arvin Rokni, Gothenburg, May 2024

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

AE	Autoencoder
CAN	Controller Area Network
CNN	Convolutional Neural Network
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DNN	Deep Neural Network
FNN	Feedforward Neural Network
FN	False Negative
FP	False Positive
HDBSCAN	Hierarchical DBSCAN
IQR	Interquartile Range
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
MSE	Mean Squared Error
Q1	First Quartile
Q3	Third Quartile
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
TN	True Negative
TP	True Positive
UMAP	Uniform Manifold Approximation and Projection for Dimension Reduction

Contents

List of Figures

List of Tables

1
Introduction

The following chapter is intended to introduce the reader to this study. Firstly a brief background and related work is given in Sections 1.1 and 1.2. Thereafter, the aim and objectives are explained in Section 1.3. Lastly, limitations and ethical considerations are presented in Sections 1.4 and 1.5, along with the thesis structure in Section 1.6.

1.1 Background

As a world-leading truck manufacturer, Volvo Trucks has developed gearbox technology for over 20 years [1]. Their automated manual transmission system uses embedded control systems to automate gear shifting efficiently based on driving conditions [2]. By being fully integrated with the rest of the truck, it incorporates numerous sensors that provide information about the vehicle.

One area of focus in recent years within Volvo and the automotive sector in general, has been to study the applicability of machine learning and neural networks to improve vehicle performance and enhancing predictive maintenance [3; 4]. For this purpose, a common technique used is anomaly detection in data analysis, which identifies patterns or instances that stand out from the norm within the data. These patterns are often referred to as anomalies, outliers or faults. Anomalies in the data originates from errors and noise or changes in system behaviour. Identifying these outliers is important for addressing errors and mitigating potential disturbances [5].

At Volvo Trucks' Driveline section, where this study is requested, valuable time is spent on classifying gear shift types and detecting abnormal data. Current way of detecting faults relies on the expertise from engineers within the field and often a statistical approach is taken. Thus, ongoing research examines the possibilities of using machine learning to enhance automation in the process making the analysis less subjective and more data-driven.

1.2 Related Work

At Volvo Group, several studies have been done in various vehicular systems using anomaly detection with machine learning and neural networks. In a case study performed by [6], a long short-term memory (LSTM) autoencoder (AE) neural network was used in order to provide a framework for detecting faults in public transport

bus air systems. Through analysis of collected multivariate time series sensor data from ten sensors (engine load, engine speed, air pressure etc.), the authors achieved to detect deviations from normal operation patterns and thereby predict system component failures. After comparing the network with a multi-layer autoencoder, the LSTM-AE showed higher performance.

Furthermore, [7] demonstrated the use of deep neural networks (DNNs) in classifying and differentiating time series data from gear shifts within Volvo's articulated haulers. Using a combination of a multivariate LSTM and a fully convolutional network, different types of gear shifts were possible to be classified with an accuracy over 90%. By detecting the shifts that deviated from what was a expected shift classification, control parameters could be adjusted in order to adapt the gear shift process.

Similarly, [8] studied the applicability of DNNs on fault detection in marine engines at Volvo Penta. The study aimed to train neural networks by using field test data in order to predict future trouble codes. Focusing on predictive maintenance, different hyperparameter settings were tested on ten feedforward neural networks (FNNs) and recurrent LSTM networks, respectively. Results showed that both network architectures are feasible to use for accurate predictive maintenance. However one LSTM network architecture obtained from configuring the hyperparameters using grid search, resulted in highest performance.

1.3 Aim and Objectives

The aim of this work is to develop a machine learning model that applies clustering- and classification algorithms in order to classify unlabeled gear shift data. By defining and evaluating different neural networks, the model further intends to be used for abnormal gear shift prediction. The potential applications of these findings intends to support engineers in analyzing large amount of data more efficiently, highlighting the most critical issues.

With the provided aim description, the following objectives are to be pursued:

1. Develop either labeling or clustering methods, or a comination of both, to classify gear shift data using unsupervised machine learning.

2. Investigate how different autoencoder neural networks can be used to effectively predict abnormal gear shifts in heavy-duty trucks.

1.4 Scope and Limitations

This thesis was done with the scope of exploring the potential in aiding engineering with data analysis of sensor outputs from heavy-duty trucks. The work therefore

does not focus on creating an machine learning model in terms of optimal performance, but instead creating a model that is functional and easy to further develop. For clustering and classification, the scope is to identify different types of gear shifts, as these were initially unknown. Furthermore, it is central to create a model where autoencoder neural networks succeed in detecting what appears to be abnormal gear shift data. The performance of the clustering algorithm and the autoencoders are important to consider, however not something that is sought to be optimized in this work.

One possible limitation in this study is that all the data processed by the machine learning model is collected from truck gearboxes. This means that other vehicle systems that could potentially be useful and provide greater insight into analyzing unregistered or anomalous events are excluded. At the same time it is reasonable to exclude other vehicle types, as the given truck data is considered to be of a sufficiently large amount to be able to develop accurate machine learning models. The work is carried out for Volvo Trucks, but it should be possible to apply the model on sensor data from other vehicle types and systems as well.

Another limitation is that the model risks lacking transparency when it comes to understanding how neural networks make specific decisions or predictions. This is a possible consequence of machine learning if it is built as a black box, where the model learns to make decisions by automatically adjusting trainable parameters based on given input but the way the parameters interact is not easily understood.

Furthermore, the given data is initially unlabelled which is a limitation since no annotated test data exists. There are no current examples of different gear shift types and there are no examples of normal or abnormal gear shifts. Because of this, it is not precisely obvious how to evaluate the performance of the machine learning model.

1.5 Ethical and Sustainability Aspects

With respect to the ethical and sustainability aspects of the study, the following offers a brief reflection on the contribution from our proposed methodology in conjunction with Volvo Group's policies.

Volvo Group's target is to have 35% fully electric sales by 2030 and their ambition is to be net-zero in their value chain by 2040 [9]. To reach their goals, Volvo Trucks evaluates the impact of their vehicles through five parameters: materials and production, fuel, exhaust emissions, maintenance and end-of-life treatment. For example, when buying a new Volvo truck, about one third of the total weight of a new vehicle is made from recycled material. One problem arises when, despite being able to use recycled materials in modern trucks, the cost of recycling is sometimes higher than the cost of extraction [10]. Therefore, it is important to have trucks that are long-lasting, leading to reduced maintenance waste and costs. Our contribution lies in the capability to detect faults in trucks, simplifying the maintenance process and

saving valuable time for engineers.

In addition, no ethical concerns are expected since only vehicular sensor data is used. A potential concern could be that external parties might use this research for their own purposes, possibly involving the use of personal data. However, considering this study is public, anyone has the right to access it.

1.6 Thesis Structure

The following list describes how the upcoming chapters are structured:

- Chapter 2 outlines the underlying theory for the statistical methods and machine learning that was applied during the work.
- Chapter 3 describes how given sensor data was collected and explored.
- Chapter 4 explains the work process used in the study, highlighting data prepossessing and methods used for gear shift type classification and abnormal gear shift prediction.
- Chapter 5 presents obtained results from gear shift type classification and abnormal gear shift prediction.
- Chapter 6 discusses the obtained results from Chapter 5, further improvements and drawn conclusions.

2
Theory

The following chapter describes the underlying theoretical principles and concepts used in this paper. Firstly diverse statistical methods such as interquartile range, data normalization and dimension reduction are explained in Section 2.1. Thereafter, an introduction to machine learning is given in Section 2.2 along with a brief overview of different artificial neural networks architectures, their learning process and performance metrics. Lastly, density-based clustering algorithms are described with a commonly used clustering evaluation metric.

2.1 Statistical Methods for Data Analysis

Statistical methods are valuable for understanding underlying structures within data, using diverse measures in order to for instance manage visualizations and data transformations. This section focuses on describing the statistical methods used in this study.

2.1.1 Interquartile Range

Interquartile range (IQR) is a technique for identifying outliers within a dataset [11]. First, data is segmented into three quartiles, which can be visualised through a box plot as shown in Figure 2.1. This plot shows a rectangular box illustrating the middle 50% of the data, with a center line denoting the median and two whiskers extending from the box representing outliers. The IQR range is the difference between the first and third quartile given by

$$\text{IQR} = Q3 - Q1, \tag{2.1}$$

where $Q1$ and $Q3$ are the first and third quartile respectively.

2.1.2 Loss Functions

Loss functions measure the difference, or loss, between a predicted model output and the ground truth. The functions are commonly used in optimization problems and machine learning where a model seeks to minimize the loss as much as possible to achieve accurate predictions. Two commonly used loss functions are the mean squared error (MSE) and the mean absolute error (MAE), as described below.

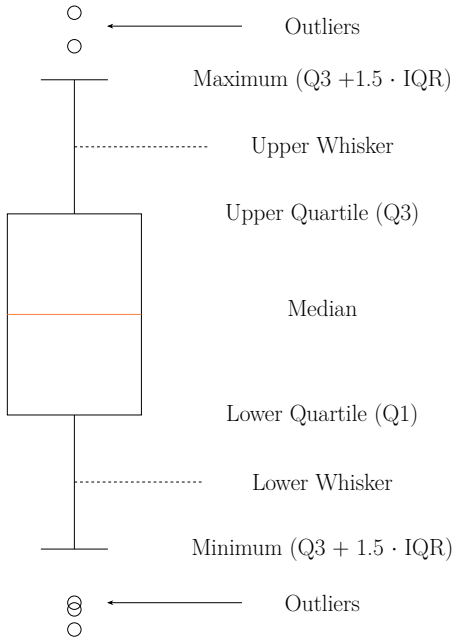


Figure 2.1: Box plot with commonly used statistical measures.

2.1.2.1 Mean Squared Error

The mean squared error is the average of the squared differences between predicted and actual values calculated as

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2, \tag{2.2}$$

where n is the number of data points, y_i represents the actual value of the ith data point, and $\hat{y}_i$ represents the predicted value of the ith data point. The metric has advantages to be used when the aim is to minimize and highlight large errors [12].

2.1.2.2 Mean Absolute Error

The mean absolute error is the average of the absolute differences between the predicted and actual values calculated as

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|, \tag{2.3}$$

where n is the number of data points, y_i represents the actual value of the ith data point, $\hat{y}_i$ represents the predicted value of the ith data point [13]. One advantage of the MAE is that it is simple to implement and it is robust to outliers [14]. Furthermore, MAE does not punish large errors as much as the MSE does, making it a suitable choice when outliers are present in the data and one wishes to minimize the influence of these.

2.1.3 Similarity Measures

Similarity measures are distance metrics that are often used in clustering and classification tasks [15]. They calculate the similarity or dissimilarity between pairs of data points and in this paper the Euclidean distance, a popular distance due to its simplicity, will be used.

2.1.3.1 Euclidean Distance

The Euclidean distance d is defined as the length of a line segment connecting two points:

$$d = \sqrt{\sum_{i=1}^{n}(x_{i2} - x_{i1})^2}. \tag{2.4}$$

Here n is the number of dimensions, and x_{i1} and x_{i2} are the coordinates of the ith dimension for the two points.

2.1.4 Data Normalization

Data normalization is a method used in data analysis to efficiently scale and organize data within a dataset, typically performed during data preprocessing. By following specific rules, normalization restructures data, making it more flexible and scalable. The process is necessary because raw data often consists of a wide range of values, which can make some machine learning algorithms to malfunction.

2.1.4.1 Min-Max Normalization

The min-max normalization is a simple normalization method that re-scales data to the range [0, 1]. Each original value x transforms into the normalized value x' according to

$$x' = \frac{x - \min(X)}{\max(X) - \min(X)}, \tag{2.5}$$

where $\min(X)$ and $\max(X)$ denotes the minimum and maximum value in the dataset X, respectively.

2.1.5 Dimension Reduction

Handling high-dimensional data can pose challenges as there is not always a straight forward method to accurately verify the meaning or correlation of the data. Hence, an effective approach is applying dimension reduction techniques, which reduces the data to a two- or three dimensional space. In this paper, the uniform manifold approximation and projection for dimension reduction (UMAP) was chosen as the dimension reduction technique.

2.1.5.1 UMAP

Uniform manifold approximation and projection for dimension reduction (UMAP) is a dimension reduction technique that is used to visualize high-dimensional data in lower dimensions [16]. First, it computes a pairwise similarity, typically the Euclidean distance between all data points in a dataset. Next, a probability distribution of the distances using a weighted curve is created, which assigns higher probabilities to pairs of points that are close to each other. The shape of the curve depends on the number of neighbours, which is the size of the local neighborhood UMAP observes. The last step is to map the higher dimensional data points onto a lower dimensional space while keeping the similarities. Using gradient descent, the technique minimizes the divergence between the probability distributions of the high-dimensional and low-dimensional data points. The final outcome is a lower-dimensional feature space of the original data, where the points represent the similarities observed in the high-dimensional space. A simple illustration of the UMAP process is shown in Figure 2.2.

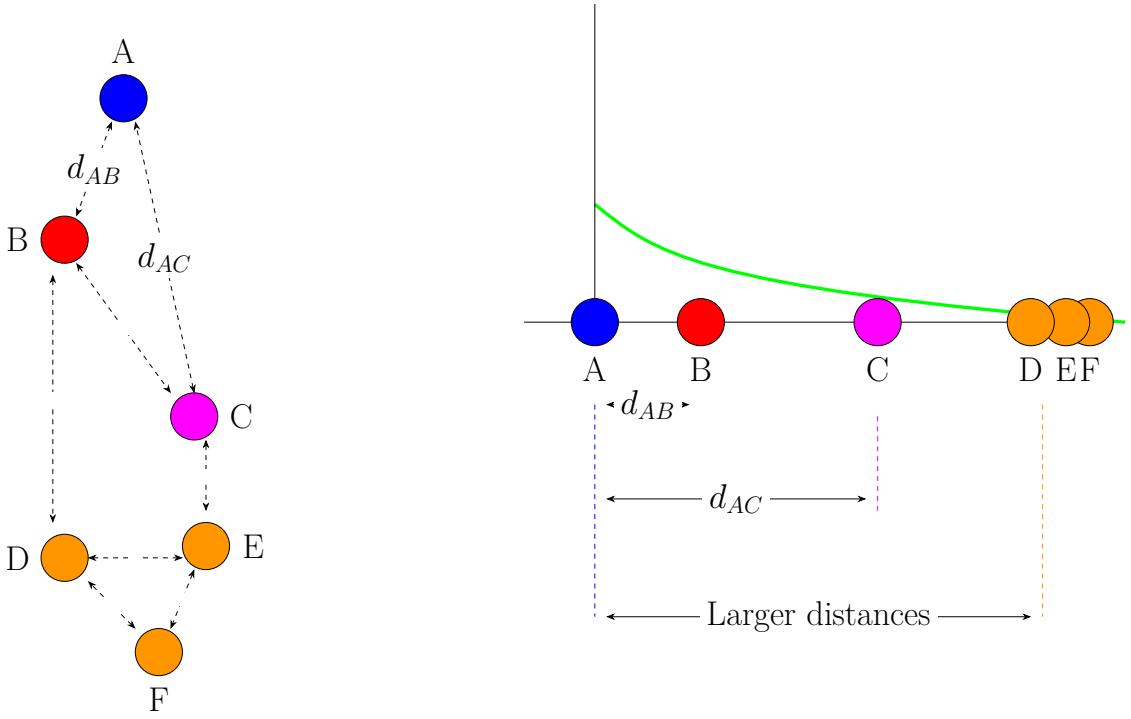


Figure 2.2: Illustration of UMAP dimension reduction technique, reducing high-dimensional data onto a lower dimension space. The left side shows how the distances between the data points are calculated. The right side demonstrates how probabilities are applied to each point relative to point A.

2.2 Machine Learning

Machine learning, described by Arthur Samuel in 1959, revolves around the concept of enabling computers to learn from experience while significantly reducing the need

for explicit programming [17]. Depending on the task to be solved and the nature of available data, there are different ways in which the learning can evolve [18]. When data samples have been identified and assigned with individual labels beforehand, the learning is said to be supervised. Otherwise the learning is considered to be unsupervised.

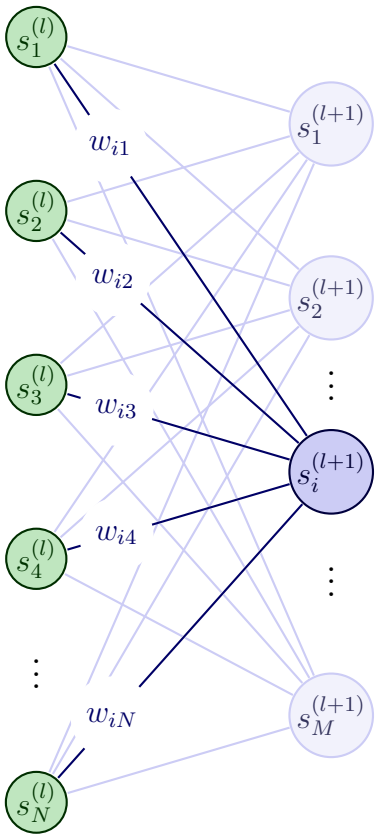
In the paper "Some Studies in Machine Learning Using the Game of Checkers" [19], Samuel described neural networks as an approach to the machine learning problem referring to prior work of Warren S. McCulloch who sought to compare the digital computer to a nervous system. Together with Walter Pitts, McCulloch performed research on nervous activity in 1943, modelling the state of a neuron as a binary threshold function [20; 21]. One neuron was assumed to be connected with other neurons through synapses propagating input information. If the input information excited the neuron above some threshold value, the neuron state would be considered active, otherwise inactive. Although highly idealized, the McCulloch-Pitts neuron is to this day still linked to the fundamental computation unit of most neural network algorithms in machine learning.

2.2.1 Artifical Neural Networks

Artificial neural networks are models within the field of machine learning that are organized in an input layer, one or more hidden layers and an output layer, where each layer is made up of artificial neurons. If a network consists of at least two hidden layers it is called a deep neural network. As illustrated in Figure 2.3, the connection strengths between neuron i in layer l and neurons $j = 1, ..., N$ in layer $l + 1$ are defined as weights w_{ij}. Neuron i computes the weighted sum $\sum_{j=1}^{N} w_{ij} s_j^{(l)}$ over the neuron states $s_j^{(l)}$ in layer l. Given a threshold value θ_i for activation of neuron i, the state of neuron i is computed as

$$s_i^{(l+1)} = g\left(\sum_{j=1}^{N} w_{ij} s_j^{(l)} - \theta_i\right). \tag{2.6}$$

Here N is the number of weights connected to neuron i and $g(b_i)$ is known as the activation function with the local field $b_i = \sum_{j=1}^{N} w_{ij} s_j^{(l)} - \theta_i$ as its argument.



$$= g\left(w_{i1}s_1^{(l)} + w_{i2}s_2^{(l)} + \ldots + w_{iN}s_N^{(l)} - \theta_i\right)$$
$$= g\left(\sum_{j=1}^{N} w_{ij}s_j^{(l)} - \theta_i\right)$$

Figure 2.3: State computation of neuron i in layer $l+1$ consisting of M number of neurons. The weighted sum $\sum_{j=1}^{N} w_{ij}s_j^{(l)}$ is over N neuron states in layer l, θ_i is a threshold for neuron i and g is a activation function.

In order to handle the effect of network instability due to function leaps that may occur at $b_i = 0$ for binary threshold functions, activation functions are typically chosen as non-linear and continuous. Examples of such activation functions that are commonly used in machine learning with neural networks are shown in Figure 2.4.

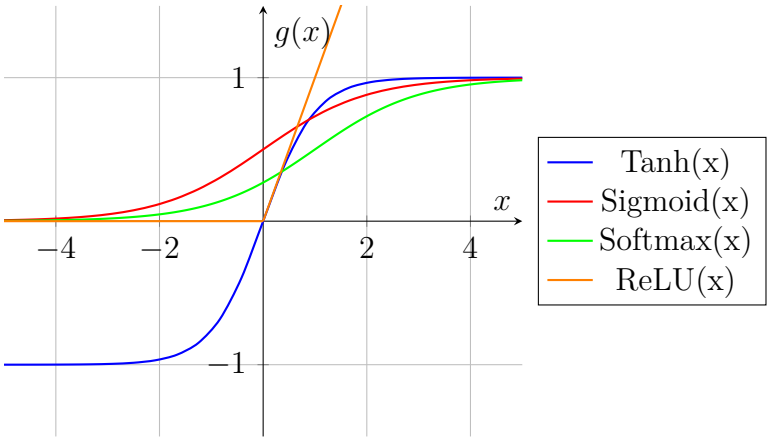


Figure 2.4: Non-linear and continuous activation functions $g(x)$, $x \in \mathbb{R}$.

2.2.1.1 Backpropagation

The learning process and the principles of how a neural network is used for machine learning to solve problems by learning from data are centered around adjusting a network's weights and thresholds. Starting from the input layer, neurons pass forward state computations layer by layer until the output layer computes a prediction. The difference between an actual output and predicted output is then quantified by a loss function which provides a measure of the network's output error. The process until now is what is known as a forward pass or forward propagation and networks that propagate data only towards the output layer are often called feedforward neural networks (Figure 2.5).

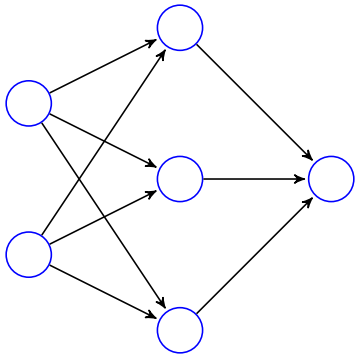


Figure 2.5: Feedforward neural network with one input layer, one hidden layer and one output layer.

The goal with the learning process is to minimize the output error computed by the output layer. This is done by iteratively computing gradients of a loss function with respect to the weights and thresholds and thereafter updating them with a defined learning rate until the loss is minimized. The update rule after iteration t is given by

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \cdot \nabla L(\mathbf{w}_t), \tag{2.7}$$

where $\mathbf{w}$ collectively denotes network weights and thresholds, η is the learning rate and L is the loss function. In mathematical terms, the learning process is an optimization problem using gradient descent on the loss function as the objective function. In contrast to the forward propagation the learning process is often called the backward pass or simply backpropagation, since it propagates the output error back through the network and the weights and thresholds are adjusted in each layer starting from the output layer and finishing at the first hidden layer [22; 23].

2.2.1.2 Hyperparameters

Hyperparameters refer to parameters within a neural network that either describe the structure of the network or the network's learning process. In Table 2.1 commonly used hyperparameters are presented. In order to optimize the performance of a neural network, hyperparameters are tuned during training. Several methods have been developed to automatically find optimal hyperparameters, whereas grid

search and random search methods are commonly used. However, when the hyperparameters are small in number, a manual search is also a common choice.

Table 2.1: Commonly used hyperparameters in a neural network.

Parameter	Description
Number of layers	Determines the depth of the neural network.
Layer dimensionality	Defines the number of neurons within each layer.
Activation function	Used to compute neuron states and introduces non-linearity into the neural network.
Epoch	Represents one complete cycle of forward pass and backpropagation.
Learning rate	Defines the step size of weight or threshold updates during training.
Batch size	Determines the number of samples processed during each epoch.
Loss function	Measures the difference between predicted and actual network output.
Optimizer	Optimization method for updating weights and thresholds based on the loss function.
Dropout rate	Rate of randomly ignored neurons during training.
Validation split	Partitioning used for setting aside a portion of the data to evaluate the neural network during training.

2.2.1.3 Autoencoders

Autoencoders are neural networks that learns to reconstruct, or simply copy input data [22; 24]. While training, autoencoders apply dimension reduction and learn to extract the most essential features that most precisely reconstructs the original input data. Since autoencoders are trained to reconstruct their own input data and are independent of labeled data to classify the reconstructions, the learning process is considered unsupervised or self-supervised, which explains the *auto* in autoencoders.

The network architecture for an autoencoder is organized in three components: an encoder, a latent space and a decoder as shown in Figure 2.6. The encoder consisting of the initial hidden layers, takes the original input data from the input layer and layer-wise reduces the data dimension until reaching the latent space where the data is compressed and the data dimension has been reduced the most. The compressed data is then gradually reconstructed through the decoder consisting of the final hidden layers that increase the data dimension until reaching the output layer which produces the reconstruction of the input data.

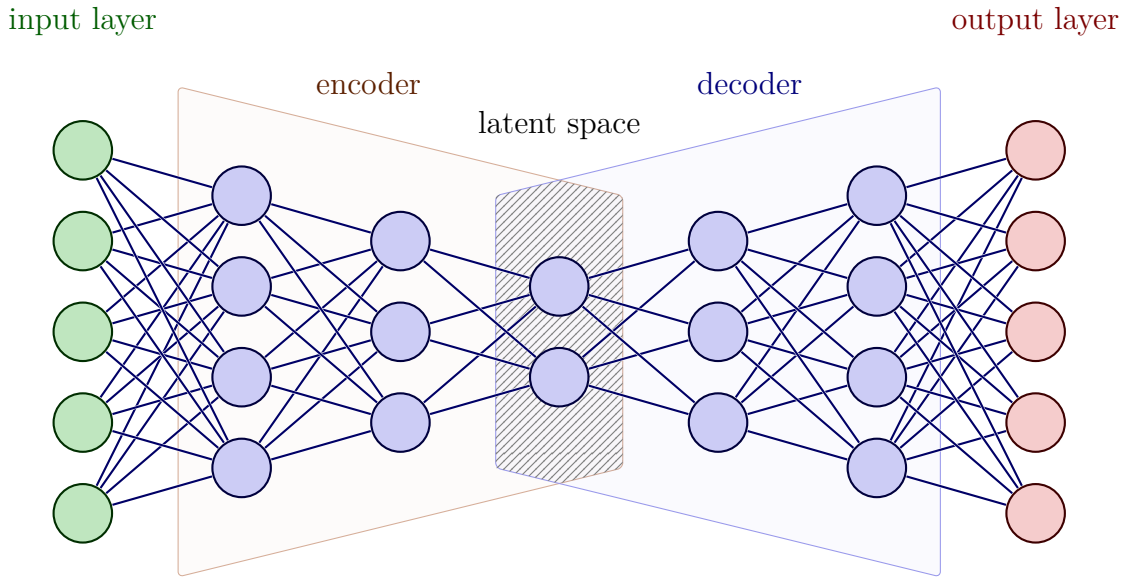


Figure 2.6: Autoencoder example architecture with an encoder, a latent space and a decoder.

Instead of using labeled data, the reconstructed data is compared to the original input data using a measure called the reconstruction error. The error measures the difference between the original input data and reconstructed data and an autoencoder that has achieved low training loss is able to capture essential features in the input data within the latent space while minimizing the reconstruction error.

2.2.1.4 Convolutional Layers

Convolutional neural networks (CNNs) are neural networks that have become the standard in machine learning when processing grid-structured data such as images or time series [22]. As shown in Figure 2.7, the data processing is centered around the mathematical operations convolution and pooling, which are normally employed in each convolutional layer. Once the input data is fed into the network's input layer, each neuron in a convolution layer apply a linear convolution between the data and a kernel. The processed data, also referred to as a feature map, is thereafter passed through a nonlinear activation function and later fed into a pooling function which reduces the feature map dimensions and makes the processed data invariant to small perturbations.

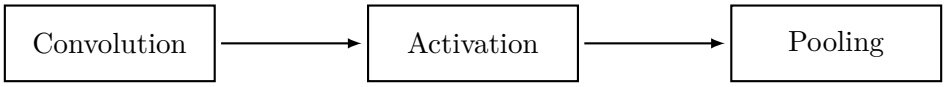


Figure 2.7: Sequence of operations in a convolutional neural network layer.

Depending on the shape of the input data, kernels and feature maps may vary in size [25]. When processing images for object recognition and detection the size is usually two-dimensional. However, for applications such as anomaly detection in power electronics and machine fault detection where the input data is a time series, the size is typically one-dimensional.

2.2.1.5 LSTM-layers

Long short-term memory (LSTM) neural networks are a type of recurrent neural networks (RNNs) that can remember information over long periods of time, capturing both short- and long-term dependencies [26]. What sets LSTM-layers apart from other types of layers is their capability of having 'memory'. As shown in Figure 2.8, the LSTM-layer consists of a cell state (C_t), a forget gate (f_t), an input gate (i_t) and an output gate (o_t) where all the gates are implemented using sigmoid activation function σ. First, the forget gate decides which parts of the previous cell state (C_{t-1}) should be forgotten based on the previous hidden state (h_{t-1}) and the current input. Next, the input gate determines which new information to store, while the tanh layer generates candidate values ($\tilde{C}_t$) to add to the cell state. The previous cell state is then updated by forgetting the specified information and incorporating the new candidate values. Lastly, the output gate determines what information of the cell state to output and scales it using a tanh function.

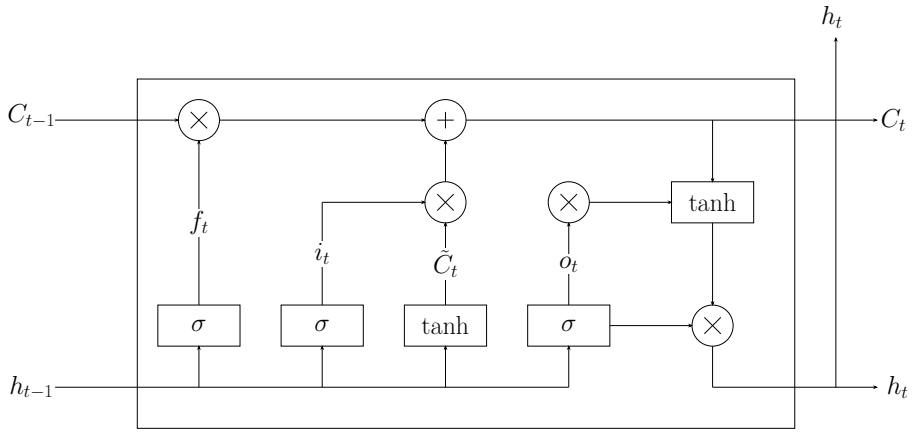


Figure 2.8: Illustration of how input data is processed through the different components of the LSTM cell.

2.2.2 Performance

Performance metrics are necessary to measure the performance of a machine learning model. Four metrics that are often used to evaluate a model are accuracy, precision and recall which are presented below [27]. Furthermore, the use of a confusion matrix in order to visualize the performance of a machine learning algorithm is explained.

2.2.2.1 Confusion Matrix

Confusion matrix is a table that is used to evaluate the performance of a machine learning algorithm when performing a classification problem. As shown in Figure 2.9, it shows the counts of true positive (TP), true negative (TN), false positive (FP) and false negative (FN) classified predictions.

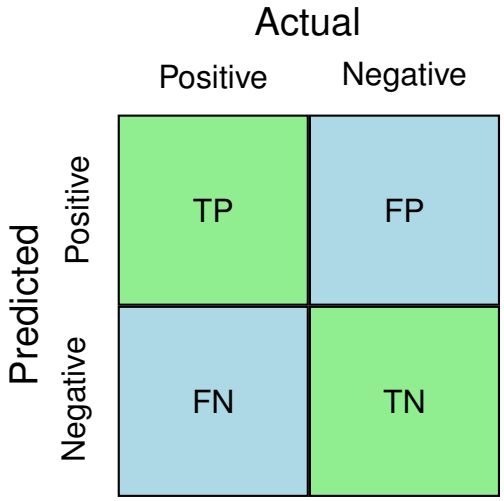


Figure 2.9: Confusion matrix for a binary classifier.

2.2.2.2 Accuracy

Accuracy, according to Equation 2.8, represents the number of correct predictions over the total number of predictions.

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \tag{2.8}$$

2.2.2.3 Precision

Precision, as expressed in Equation 2.9, is a metric that measures the ratio of true positive instances over the total positives that the model predicts. A precision score of 1 means that the model predicted all true positives, and a low precision score means the model has high number of false positives.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2.9}$$

2.2.2.4 Recall

Recall, as defined by Equation 2.10, represents the number of true positive instances over the total number of correct predictions. A recall score of 1 means that the model predicted all true positives without any false negatives.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2.10}$$

2.2.3 Clustering

Clustering is one of the roles that unsupervised machine learning algorithms have in depicting dissimilar data in different clusters and grouping similar data in the same cluster [22]. Depending on the data structure to be clustered, different similarity measures are used such as the Euclidean distance. Furthermore, clusters can be defined through different data properties and in this study clusters were based on high point density.

2.2.3.1 Density-based Clustering

Density-based clustering is a category of clustering algorithms that group data points into clusters based on areas of higher density in the feature space [28]. One advantage of density-based clustering is that it does not require specifying the number of clusters in advance. One of the most popular density-based clustering methods is density-based spatial clustering of applications with noise (DBSCAN). The main parameters that define the density-based clustering algorithms are ϵ and $MinPts$. As visualized in Figure 2.10, ϵ is the maximum distance between two points for them to be considered as part of the cluster while $MinPts$ defines the minimum number of points required to form a dense region. The algorithm works by classifying points as a core point, border point or noise. Core points defines the cluster and are the points within a distance of ϵ and have a minimum value of $MinPts$ points including itself, while border points are the points within the ϵ distance of a core point. The rest of the points are considered noise.

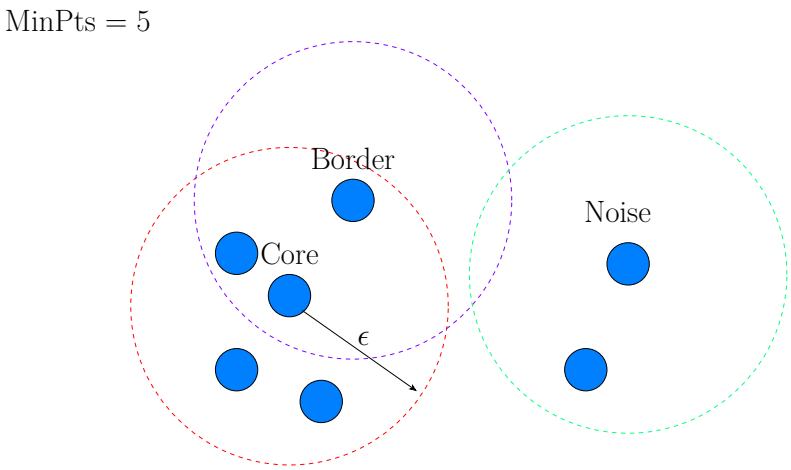


Figure 2.10: Defining of core, border and noise points in density-based clustering where $MinPts$ is set to '5'.

2.2.3.2 DBSCAN

Initially, DBSCAN selects a random point that has not been visited [29]. Then, all points that are density-reachable from this selected point, according to the parameters ϵ and $MinPts$, are marked as part of the same cluster. This continues until all points have been visited and assigned to clusters or marked as noise.

2.2.3.3 HDBSCAN

Hierarchical DBSCAN (HDBSCAN) extends the DBSCAN algorithm by converting it into a hierarchical clustering algorithm, and then using a technique to extract a flat clustering based in the stability of clusters [30]. One advantage of HBDSCAN compared to DBSCAN is that it calculates the most stable ϵ by itself.

2.2.3.4 Clustering Evaluation Metrics

In order to measure the performance of a clustering algorithm in case the ground truth is unknown, chosen evaluation metrics must be applied on the algorithm itself [31]. Below is a description of one such metric called the silhouette coefficient, which is used to measure how well defined and separated clusters are.

Silhouette Coefficient

The silhouette coefficient for a set of clusters is the mean silhouette coefficient for each sample in the set given by

$$s = \frac{a - b}{\max(a, b)}. \tag{2.11}$$

Here a is the average distance from a given sample to all other samples within the same cluster and b is the average distance from a sample to all samples in the closest neighboring cluster. The silhouette coefficient takes values between -1 and 1, where a higher value indicate better clustering and a value around 0 indicate overlapping clusters.

3
Data Collection and Exploration

The following chapter describes the origin of the provided data and the data collection process that made this work possible through software used at Volvo Trucks. Furthermore, the chapter explains performed data exploration in order to better understand the structure and properties of the given data.

3.1 Data Collection Process

The data in this study originates from sensors in different types of heavy-duty trucks and for this work, seven sensors relevant to the gearbox was given by supervisors at Volvo. The output signals were collected from field test data within Volvo and from customers where vehicles have been driven in different driving environments. Signals come from various vehicle components and are connected to a CAN-bus system [32] that is used with the CANalyzer software [33] for the purpose of monitoring and diagnosing component behaviors in a vehicle. A simplified illustration of the data collection process, from the truck to the CAN system to the computer, is shown in Figure 3.1. Throughout this paper, the gearbox was the component of the truck in focus and therefore only sensors related to the gearbox's behavior were considered. Since the provided sensor outputs were logged during different time intervals, each interval of sensor outputs is regarded as a multivariate time series where the dimension is determined by the number of sensors. Each recorded time series representing the gearbox behavior during a specific gear shift is further referred to as a log.

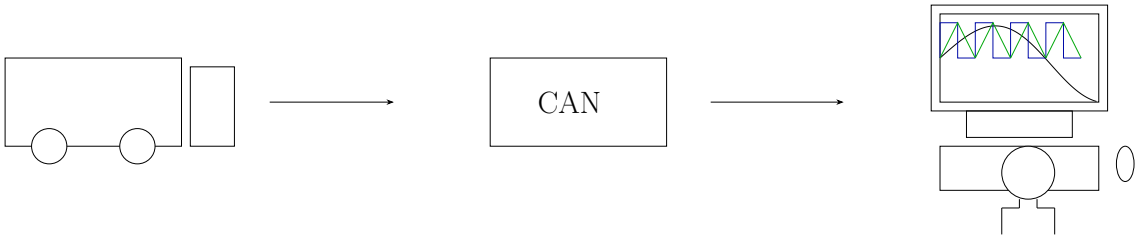


Figure 3.1: Visualization of the data collection process at Volvo Trucks.

3.2 Data Exploration

The given dataset consists of logs that are separated into different folders where each folder is a vehicle type. Some folders contain more logs than others, meaning

some types of shifts will be more prevalent while other less so. This imbalance can cause problems if not taken into consideration before training the machine learning models. As visualized in Figure 3.2a, all logs in the given dataset vary in lengths, where Figure 3.2b show that most log lengths are within a time interval of about 50 to 200 ms. Some logs appears to be shorter than a single shift while others are much longer and consists of multiple shifts, which is seen in Figures 3.3a and 3.3b. However, the majority of the logs are long enough to show one single shift. Although in the minority, there are some logs that are completely empty and others that consists of missing values, which is a misleading representation of the data and can pose challenges if not handled properly.

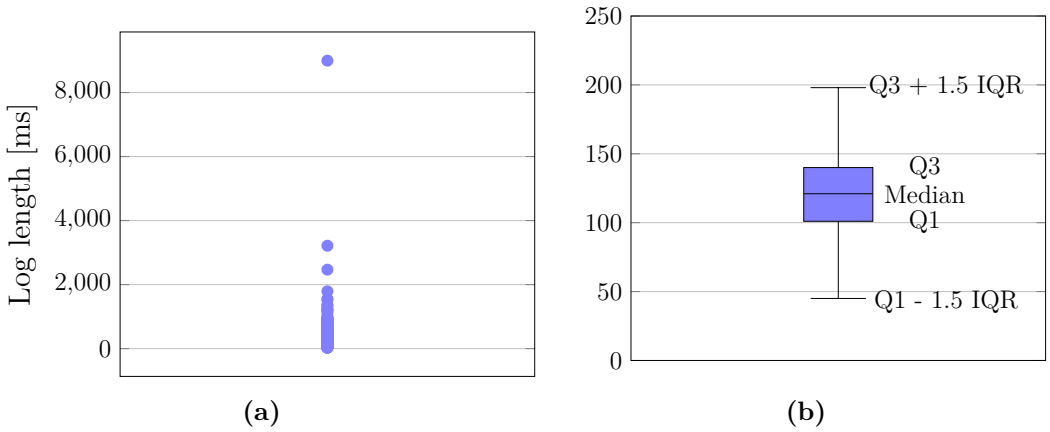


Figure 3.2: **(a)** Distribution of log lengths. **(b)** Box plot segmentation of the given sensor data.

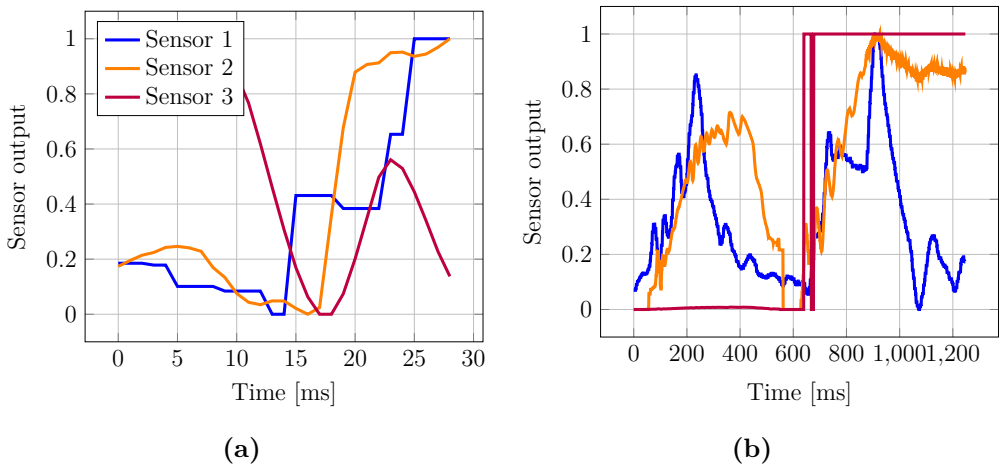


Figure 3.3: **(a)** Example log with length below the lower threshold value Q1 - 1.5·IQR. **(b)** Example log with length above the higher threshold value Q3 + 1.5·IQR.

4

Methods

The following chapter starts with explaining the overall work process used in order to implement the different methods in Section 4.1. The methods used to preprocess and classify the unlabeled sensor data from Volvo Trucks are described in Sections 4.2 and 4.3. Furthermore, the implementation and evaluation of the autoencoder architectures used to detect abnormal gear shift logs are explained in Sections 4.4 and 4.5.

4.1 Work Process

The work process was separated into different blocks for easier understanding of each step of the process. It started with preprocessing the raw sensor data by converting it to suitable format for the autoencoders. Next, the processed data underwent clustering and classification to separate training logs for each gear shift type. Once all logs were labeled, the autoencoders were trained and tested on the different clusters. After training, a threshold for the reconstruction error was set which was used to separate normal and abnormal logs in the test data. Every step of the workflow is visualized in Figure 4.1.

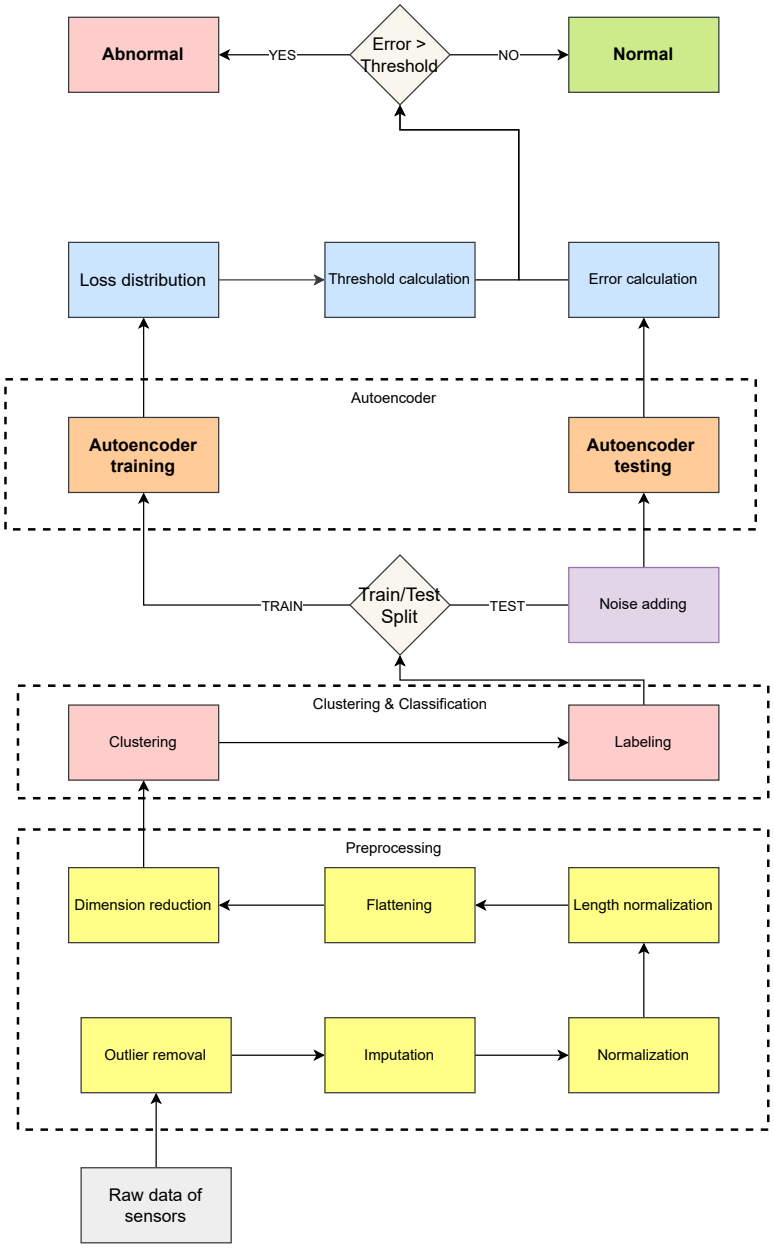


Figure 4.1: Flowchart illustrating the entire work process used in this study.

4.2 Data Preprocessing

Preprocessing had an important role in this paper, as the dataset at hand was not inherently prepared as input for the autoencoders. The primary objective of the preprocessing was to clean the data of inconsistencies and to shape it to a suitable

format.

4.2.1 Outlier Removal

In the dataset, we encountered logs of varying lengths, some too short to represent a full shift, while others were too long, containing multiple shifts. To address this, Interquartile Range (IQR) method was used to filter out outlier logs, ensuring more consistent dataset for analysis. All log lengths inside the IQR range were considered suitable lengths for the problem and everything outside was filtered out.

4.2.2 Imputation

When detecting faults in a time series it is important to identify and handle all the missing values in the dataset to prevent potential inaccuracies in the model's performance. To fill the missing values, the data was imputed using linear interpolation. It fills the missing values by connecting the available data points with straight lines, effectively estimating what the values in between the points should be [34]. This is a simple technique which maintains the shape of the signals while filling in the gaps.

4.2.3 Sensor Output Normalization

The next step in the data preprocessing was to normalize the sensor outputs from the gear shift logs. In this paper, min-max normalization was used to scale the sensor values to the range [0, 1]], as visualized in Figure 4.2.

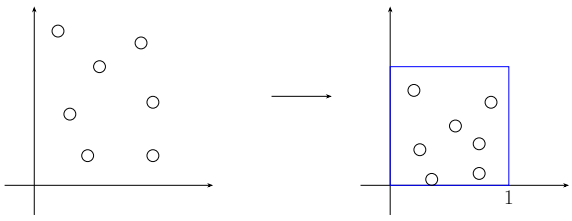
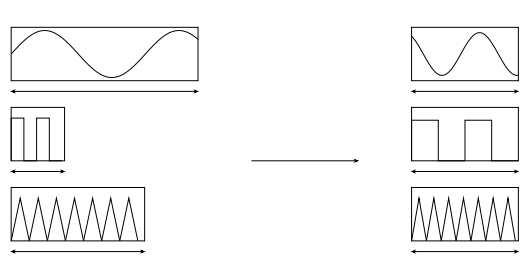


Figure 4.2: Transformation of raw data (left) into a normalized format (right).

4.2.4 Length Normalization

To simplify the data analysis, all logs were reshaped into a series and then concatenated into a single dataframe. As illustrated in Figure 4.3b, every row is a timestep and each column a sensor, where each cell corresponds to a time series. Once the logs had been concatenated, the focus shifted towards normalizing the length of each log. To achieve this, the given data was interpolated to the mean log length, ensuring that the autoencoders recieved a uniform input size. A simple visualization of the length normalization is shown in Figure 4.3a.



(a)

	Sensor 1	···	Sensor n
Log 1	0,1,2...t	···	Series
...	...	...	...
Log n	Series	···	Series

(b)

Figure 4.3: **(a)** Length normalization of gear shift logs from the given field test data. **(b)** Dataframe containing cells with sensor outputs. Each column represents a sensor and each row represents a specific log.

4.2.5 Flattening

Next step was to reshape the multivariate dataset into a univariate format, as shown in Figure 4.4. Univariate timeseries, in comparison to multivariate timeseries, are easier to work with when training and testing a machine learning model. All logs were flattened which resulted in a single time series with multiple signals from all the sensors. Since the log lengths had already been standardized in the previous step, the new time series with multiple signals also had the same lengths. After all these preprocessing steps, the dataset was usable as an input for the autoencoders.

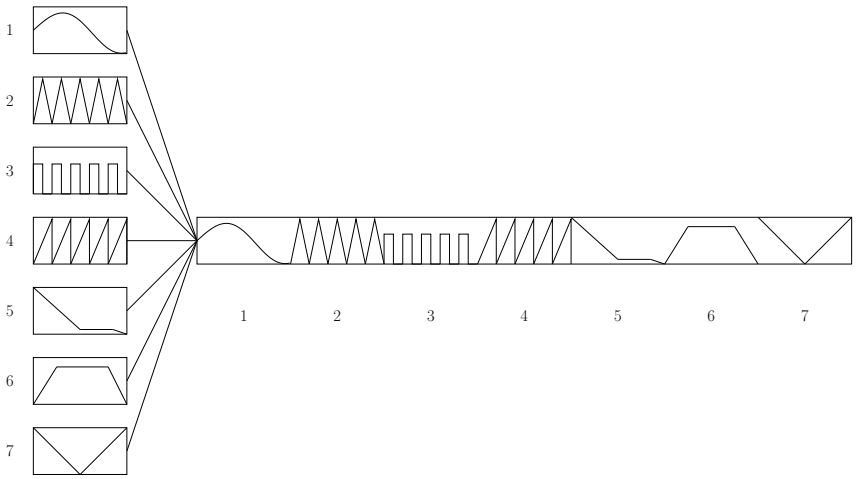


Figure 4.4: Flattening of logs with multiple sensor outputs, resulting in a univariate time series log.

4.2.6 Dimension Reduction

The final step of the preprocessing involved reducing the high-dimensional data to a two-dimensional space in order to visualize the potential clusters. Since the data given was unlabeled, it was important to visualize and verify whether there were clusters or not. For this step, UMAP was chosen as the dimension reduction technique due to its ability to preserve both local and global structures in the

data while efficiently capturing non-linear relationships. By setting the parameter *n_components* to '2', the high-dimensional data was reduced to two dimensions. Different values for the parameter *n_neighbours* were tested until distinct clusters were formed in the feature space. The default value of '15' formed distinct clusters and was kept as the chosen parameter.

4.3 Data Classification

In order to achieve the first objective of this study and identify abnormal logs within specific types of gear shifts, data classification was needed. Although autoencoders are able to reconstruct different log types simultaneously, it was decided to first classify unique types of logs before feeding them to the autoencoders which would predict abnormal logs within each log type. For this purpose, different clustering algorithms were first studied and applied on the preprocessed data. Thereafter, gear shift logs were classified with the CNNClassifier provided by the Python library sktime [35].

4.3.1 HDBSCAN

To cluster the logs using the reduced data, the built-in HDBSCAN clustering algorithm from the Scikit-learn Python library [31] was implemented. The algorithm was dependent on setting the parameter *min samples*, which was set by testing different values until distinct clusters were formed. To evaluate the clustering, other than visually inspecting the results, silhouette coefficient was used as an evaluation metric for verification.

4.3.2 Gear Shift Type Labeling

After the clustering was done and the gear shift logs were divided into separate clusters, the clusters were used to train the CNNClassifier to classify specific log types belonging to specific clusters. The training process was done using the networks default parameters from sktime according to Table 4.1. Lastly, the trained CNNClassifier was used to label all logs according to the classifier predictions.

Table 4.1: Parameter values used in the training process of the CNNClassifier.

Parameter	Value
Activation function	Softmax
Average pool size	3
Batch size	16
Callbacks	None
Kernel size	7
Loss function	Categorical crossentropy
Metrics	None
Number of convolutional layers	2
Number of epochs	50
Use bias	True

4.4 Model Implementation

In this section, the implementations of the dense, CNN- and LSTM autoencoder arcitechtures are described. Every autoencoder was implemented using TensorFlow's Keras API, with corresponding classes and parameters according to Table 4.2. All the autoencoders where trained in a similar way where the *EarlyStopping* callback function in Keras was used to monitor the validation loss and stop the training if the loss did not decrease during five successive epochs.

Table 4.2: General parameter values used in the training process of all autoencoders.

Parameter	Value
Optimizer	Adam
Learning rate	0.001
Loss function	MSE
Batch size	128
Number of epochs	100
Validation split	20%

4.4.1 Dense Autoencoder

The dense autoencoder used in this study according to Figure 4.5, consisted of an input layer, a hidden layer and an output layer in order to reconstruct the gear shift logs. The input layer consisted of 408 neurons matching the length of a flattened data log, the dense hidden layer consisted of 32 neurons with ReLU activation function and the output layer with the same number of neurons as the input layer applied the sigmoid activation function.

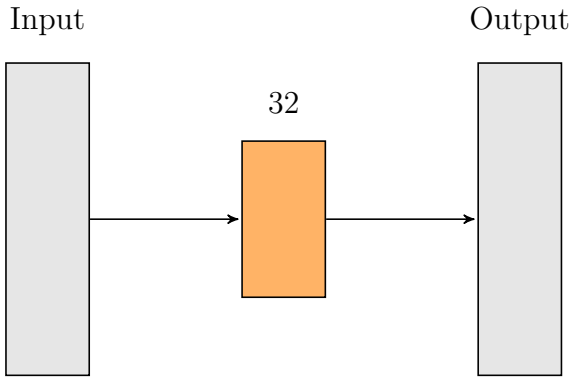


Figure 4.5: Dense autoencoder neural network with one hidden layer consisting of 32 neurons.

4.4.2 CNN Autoencoder

The architecture of the CNN autoencoder, as illustrated by Figure 4.6, consisted of six layers, where the first input layer had the input shape (408,1). The second layer was a one-dimensional convolutional (Conv1D) layer with sixteen filters and a kernel size of seven. This followed by a second Conv1D layer, which had eight filters and a kernel size of three. The decoder part of the autoencoder started with a one-dimensional transposed convolutional (Conv1DTranspose) layer that had eight filters and a kernel size of three. After that was another Conv1DTranspose layer with sixteen filters and a kernel size of seven. The sixth and last output layer was a Conv1DTranspose layer with only one filter and a kernel size of seven. The *padding* parameter was set to 'same' for all convolutional layers to ensure the same input and output shape. The output layer contained only one filter, *stride* set to '1' and a kernel size of seven. All layers, except for the input and output layers had ReLU as the activation function.

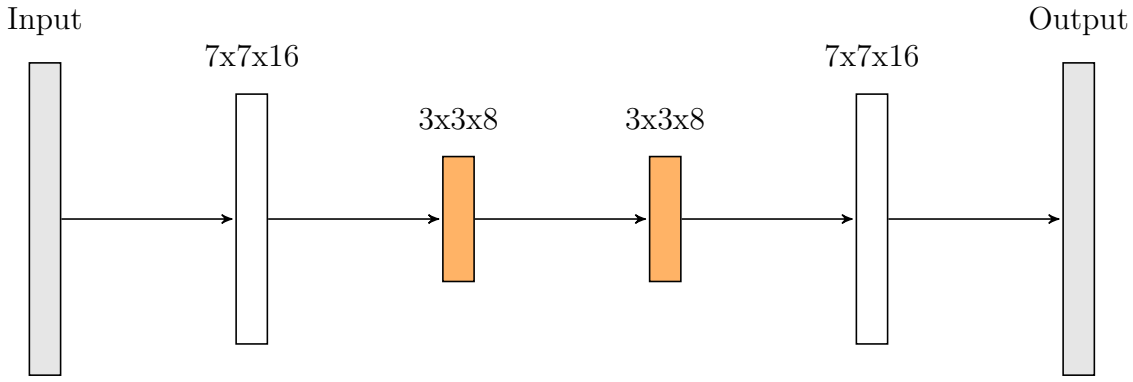


Figure 4.6: CNN autoencoder architecture with four convolutional hidden layers.

4.4.3 LSTM Autoencoder

The LSTM autoencoder architecture consisted of an input layer of size (408,1), followed by four LSTM-layers. As shown by Figure 4.7, the first layer has 16 units,

the second and third layers have 8 units each, and the fourth layer has 16 units. Lastly, a dense layer was used as the output layer with the size (408,1). All the layers in the LSTM-AE had tanh as activation function.

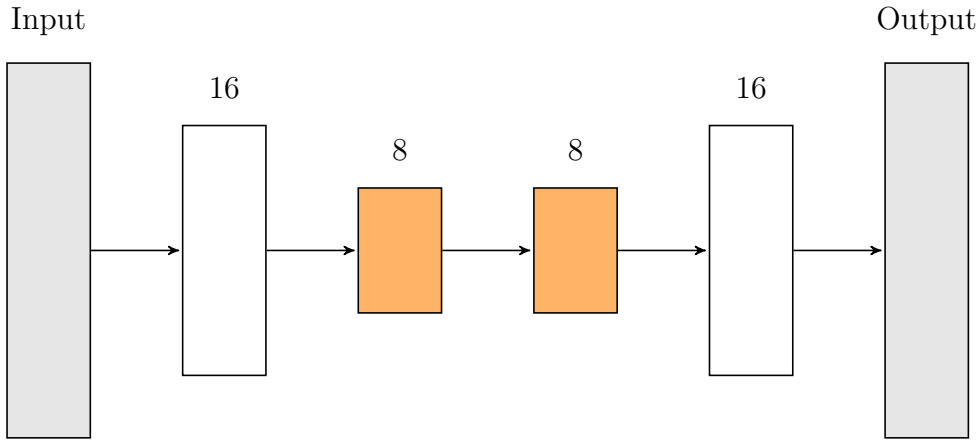


Figure 4.7: LSTM autoencoder architecture with four hidden LSTM-layers.

4.5 Model Evaluation

To evaluate the performance of the autoencoders, a self-generated test set was constructed based on the dense AE since it took least time to train. This was done by first extracting 20% of the original flattened logs into a test set, while keeping the remaining 80% for training the autoencoders. With the assumption that a vast majority (above 90%) of the original logs were normal gear shifts, 50% of the test logs were labeled as normal. For the remaining 50%, Gaussian distributed noise was added and these logs were labeled abnormal. The distribution was centered at zero with a standard deviation set experimentally by testing different values until a notable separation between the distribution of reconstruction errors from normal and abnormal test logs was observed.

5

Results

The following chapter presents obtained results in this thesis. Section 5.1 focuses on results from applied clustering algorithm and results from data labeling, while Section 5.2 presents results from implemented autoencoders when predicting abnormal gear shifts.

5.1 Gear Shift Type Classification

The following describes obtained results from the clustering and classification processes. Section 5.1.1 shows results from applying HDBSCAN and Section 5.1.2 demonstrates the results from labeling gear shift log types, where Figure 5.4 provides an overview of the distribution of logs across all found clusters.

5.1.1 Clustering

The resulting plot from the HDBSCAN clustering is presented in Figure 5.1. Each data point is coloured based on its defined cluster where the red points are considered noise. Figure 5.2 shows the remaining distinct clusters in feature space after removal of the noisy points. After calculating the silhouette coefficient, a score of 0.516 was achieved.

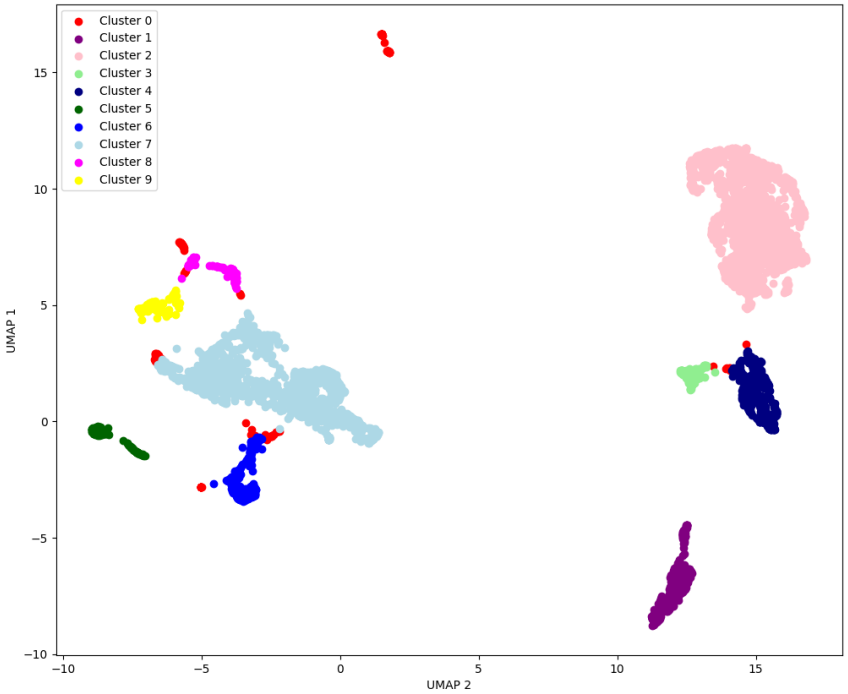


Figure 5.1: Obtained clusters from HDBSCAN, where cluster 0 represents noise.

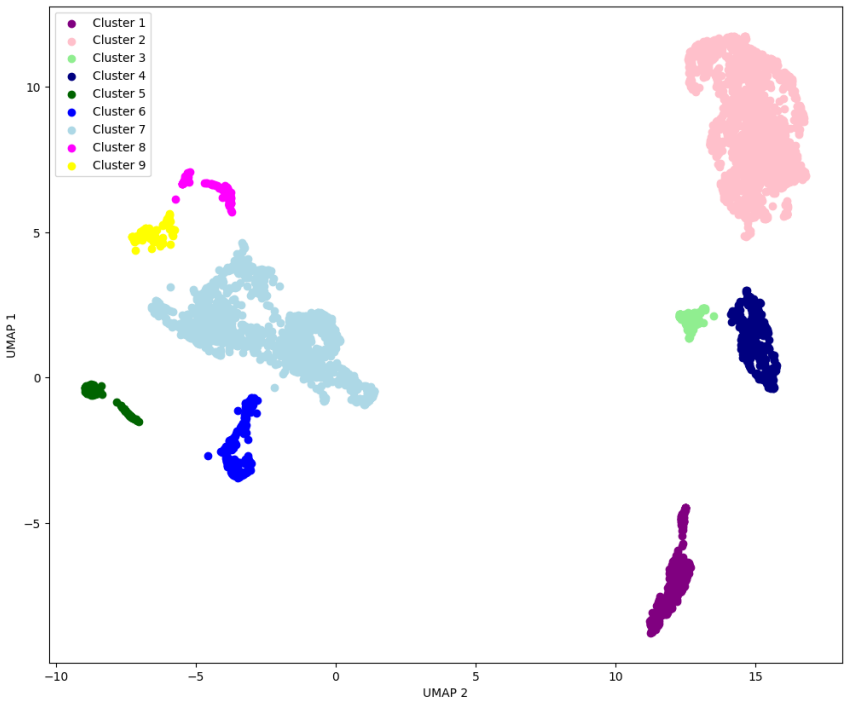


Figure 5.2: Obtained clusters from HDBSCAN after removal of noise.

5.1.2 Labeling

In Figure 5.3, the confusion matrix illustrates the performance of the CNNClassifier on the test data. The color intensity of the cells corresponds to the number of logs,

with darker cells indicating a higher number of logs. After calculating the ratio of correctly predicted logs to the total number of test logs, the classifier achieved an overall accuracy of 98%.

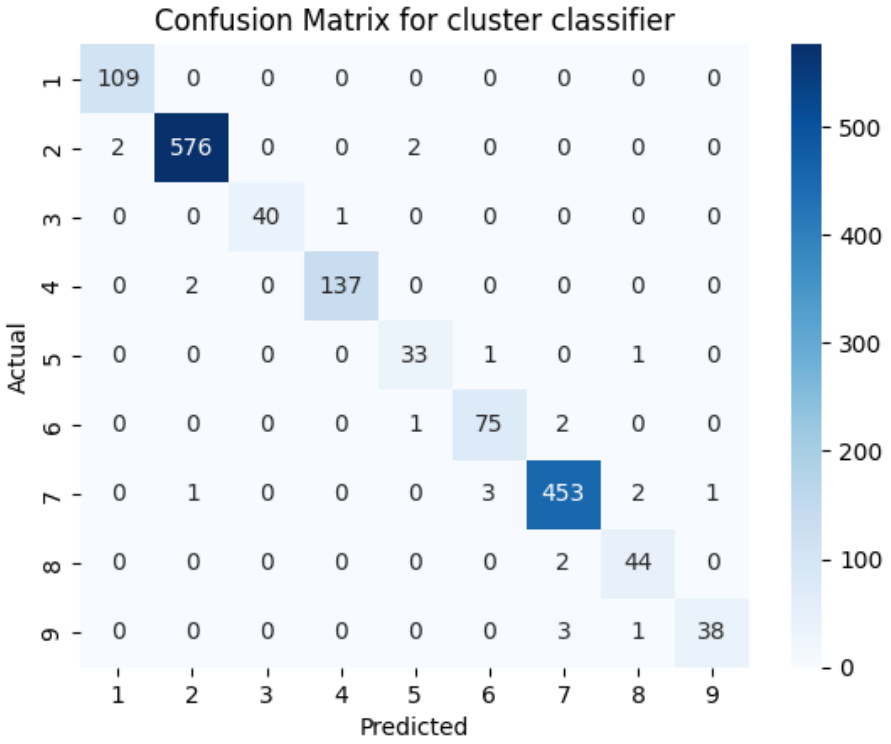


Figure 5.3: Obtained confusion matrix for the CNNClassifier after gear shift type predictions.

After labeling all the gear shift logs using the trained CNNClassifier, Figure 5.4 shows the obtained distribution of gear shift type within the nine clusters. Most logs were classified as belonging to cluster 2 and 7, while the least number of logs were classified as belonging to cluster 3.

Furthermore, Figure 5.5 shows gear shift logs within the obtained clusters, where each corresponding subfigure shows three example logs. In order to make the presentation clear, only output values from three out of the seven provided sensors are shown.

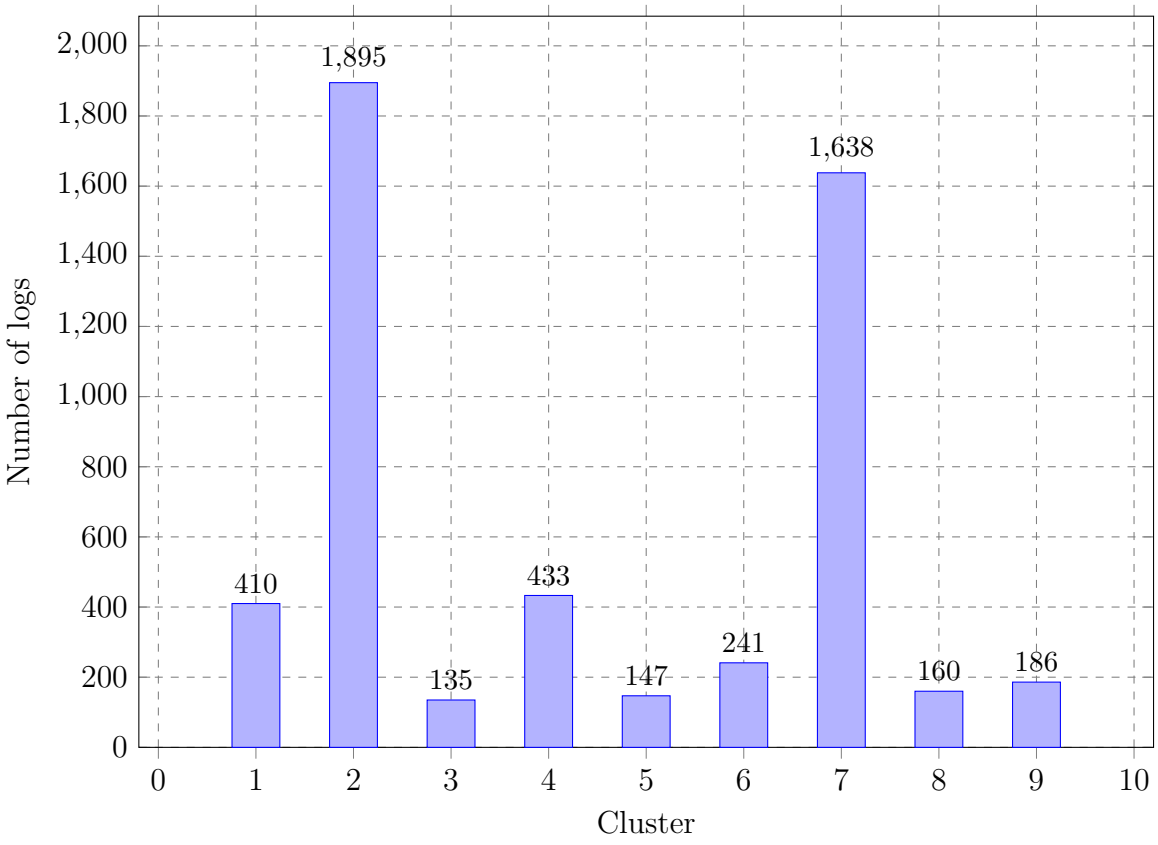


Figure 5.4: Gear shift type distribution as result of applying the CNNClassifier on all logs in the provided data.

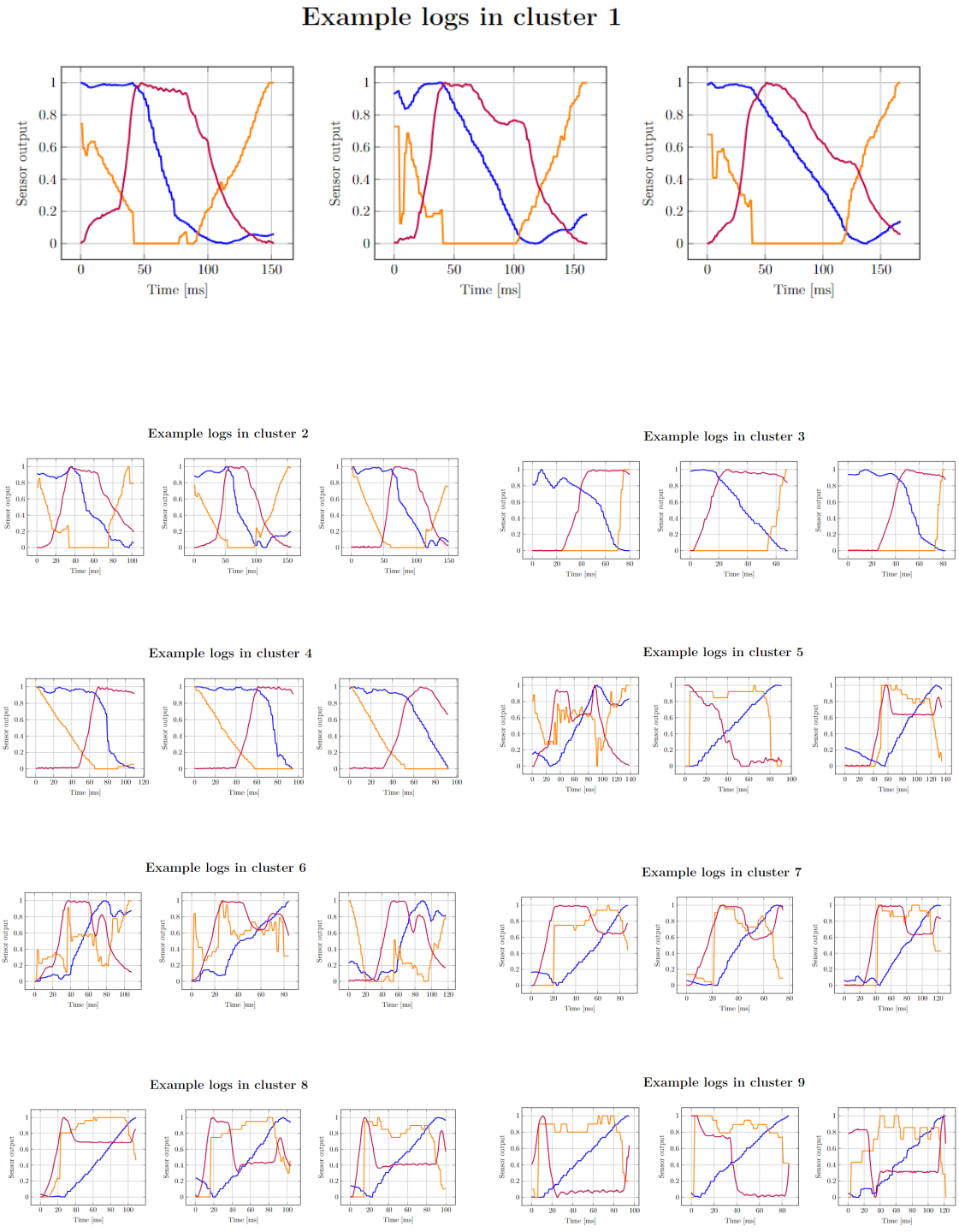


Figure 5.5: Randomly sampled gear shift logs from predicted clusters 1-9. Each log represents sensor outputs from three sensors during a gear shift.

5.2 Abnormal Gear Shift Prediction

Given the large number of gear shift logs analyzed across different clusters, this section focuses on presenting results from only cluster 1. Other clusters have been processed similarly as described in Section 4.2 but corrensponding results are not included in this section to keep the presentation concise. Results for clusters 2-9 for each autoencoder architecture are presented in Appendix A. Section 5.2.1 presents the results from training the autoencoders to reconstruct gear shift logs from the train set, while Section 5.2.2 demonstrates the performance when predicting abnormal gear shifts within the self-generated test set.

5.2.1 Training Results

The training results of the different autoencoders is presented through learning curves (Figures 5.6, 5.9, 5.12), which show the training and validation losses over number of epochs. Furthermore, the distribution of reconstruction errors of the training logs of the autoencoders is visualized through histograms according to Figures 5.7, 5.10, and 5.13. To provide further insight into the performance of the autoencoders, gear shift logs with the four smallest, four middle-sized and four largest reconstruction errors are presented in Figures 5.8, 5.11 and 5.14.

5.2.1.1 Dense Autoencoder

The following Figures 5.6, 5.7 and 5.8 represent training results from the dense autoencoder when applied to gear shift logs within cluster 1.

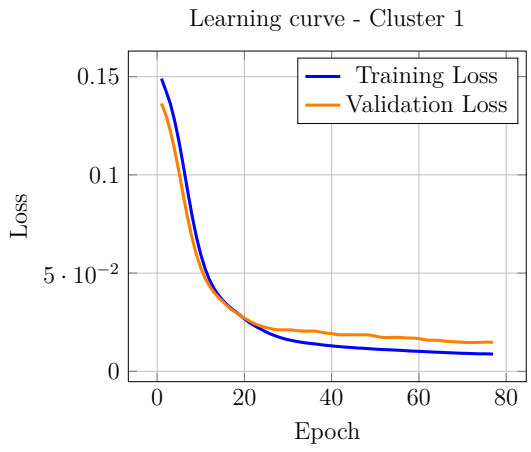


Figure 5.6: Dense autoencoder learning curve for training logs in cluster 1.

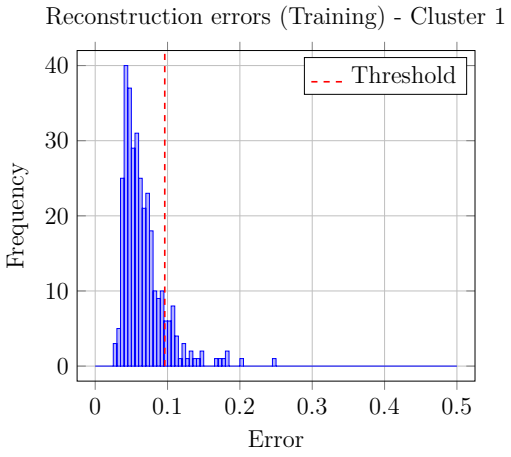


Figure 5.7: Reconstruction error distribution for training logs in cluster 1.

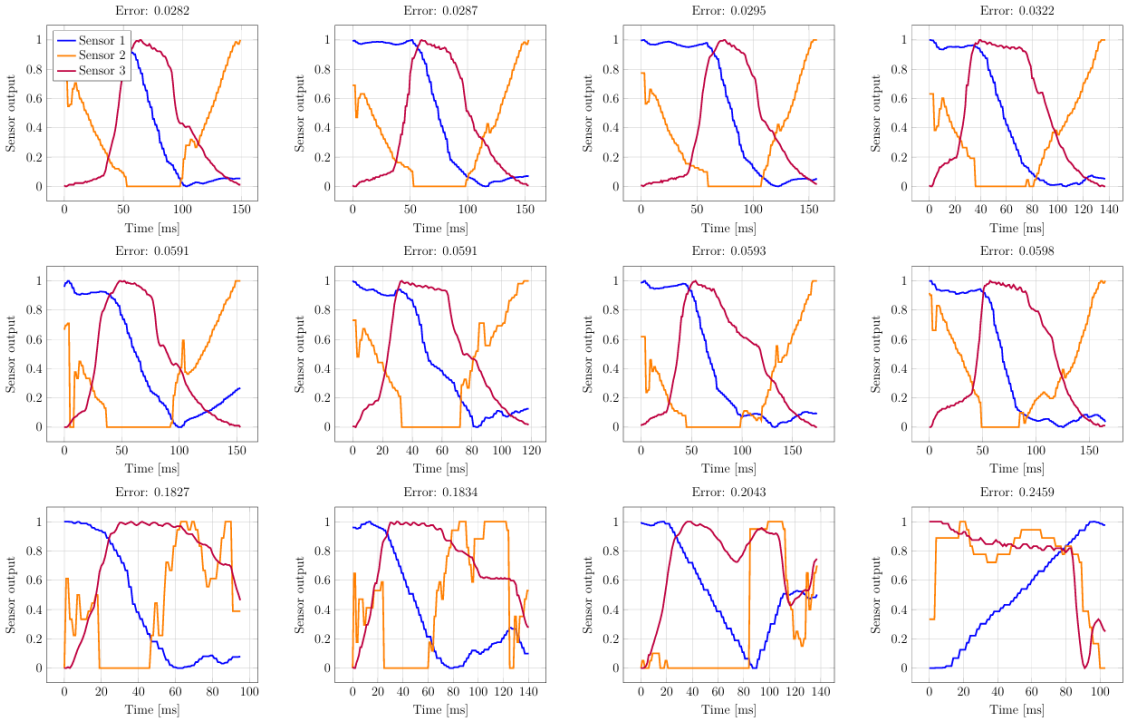


Figure 5.8: Gear shift logs in cluster 1 sorted according to their corresponding reconstruction error after applying the dense autoencoder.

5.2.1.2 CNN Autoencoder

The following Figures 5.9, 5.10 and 5.11 represent training results from the CNN autoencoder when applied to gear shift logs within cluster 1.

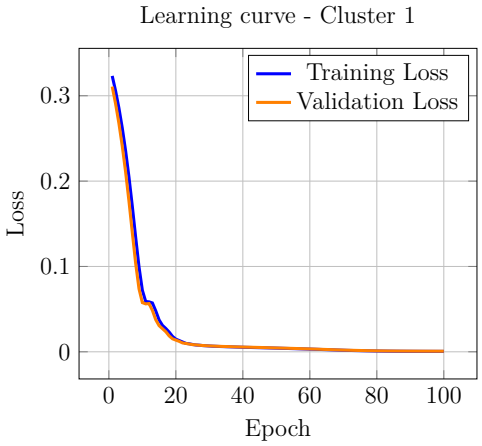


Figure 5.9: CNN autoencoder learning curve for training logs in cluster 1.

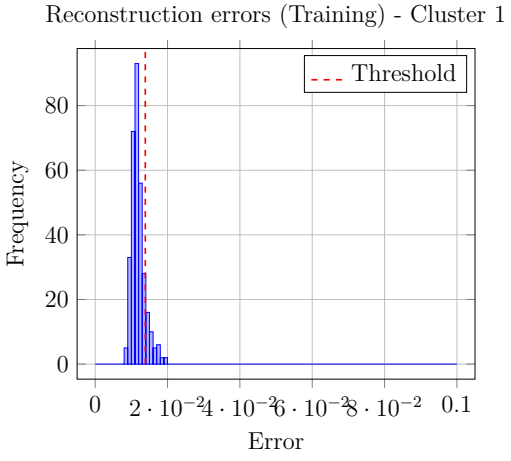


Figure 5.10: Reconstruction error distribution for training logs in cluster 1.

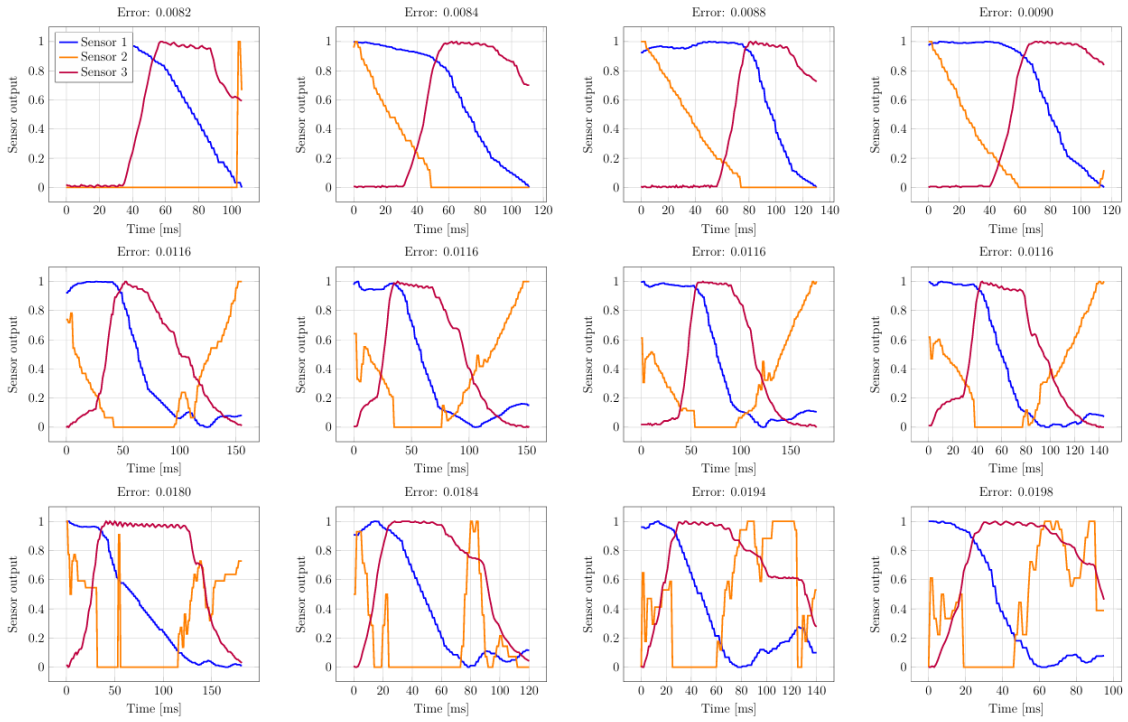


Figure 5.11: Gear shift logs in cluster 1 sorted according to their corresponding reconstruction error after applying the CNN autoencoder.

5.2.1.3 LSTM Autoencoder

The following Figures 5.12, 5.13 and 5.14 represent training results from the LSTM autoencoder when applied to gear shift logs within cluster 1.

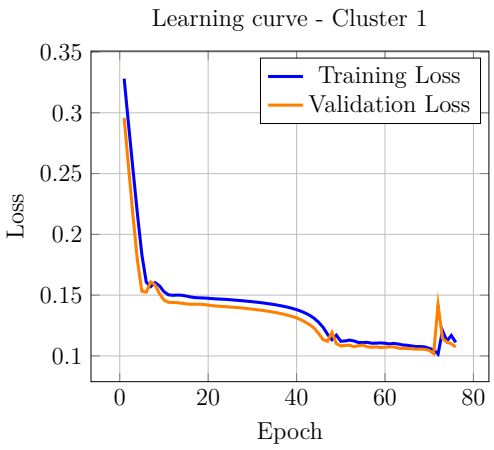


Figure 5.12: LSTM autoencoder learning curve for training logs in cluster 1.

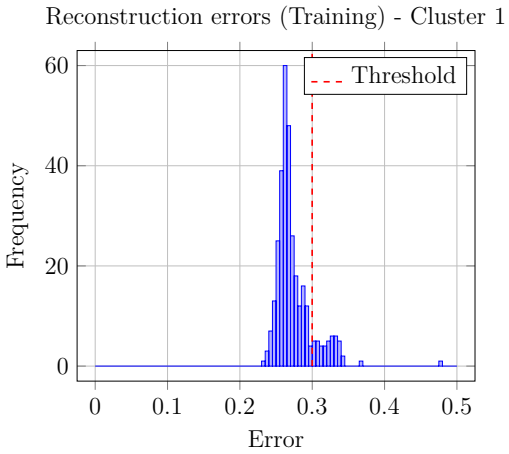


Figure 5.13: Reconstruction error distribution for training logs in cluster 1.

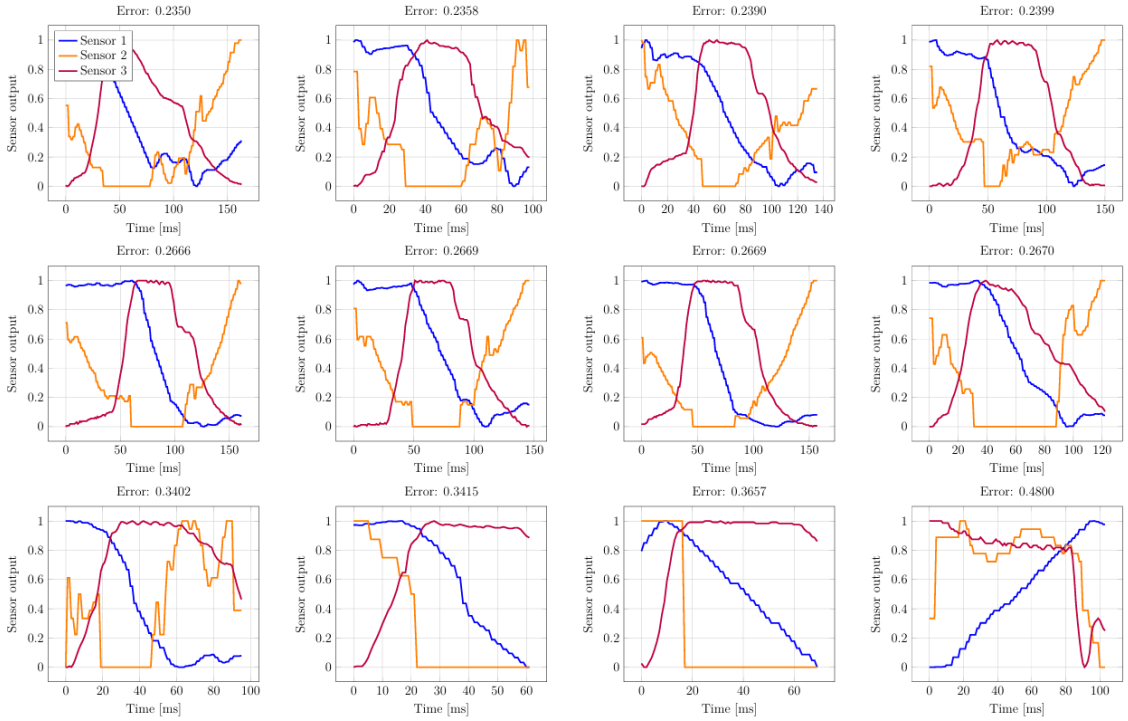


Figure 5.14: Gear shift logs in cluster 1 sorted according to their corresponding reconstruction error after applying the LSTM autoencoder.

5.2.2 Evaluation Results

For evaluation of the autoencoders, Figures 5.15, 5.16 and 5.17 illustrate the distribution of reconstruction errors for normal and abnormal test logs. Additionally, Tables 5.1, 5.2 and 5.3 present the performance of each autoencoder on the test logs with added noise as described in Section 4.5.

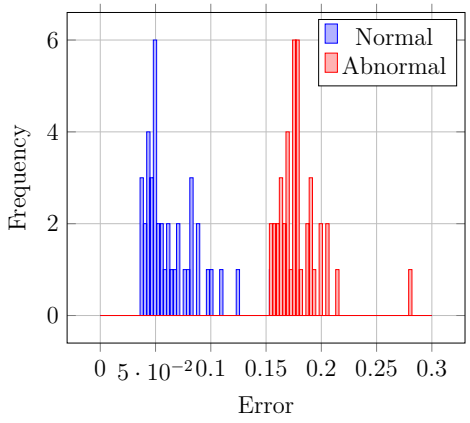


Figure 5.15: Obtained reconstruction error distributions for normal and abnormal test logs respectively after applied to the dense AE within cluster 1.

Metric	Score
Accuracy	0.939
Precision	1.0
Recall	0.878

Table 5.1: Dense AE performance on test logs in cluster 1.

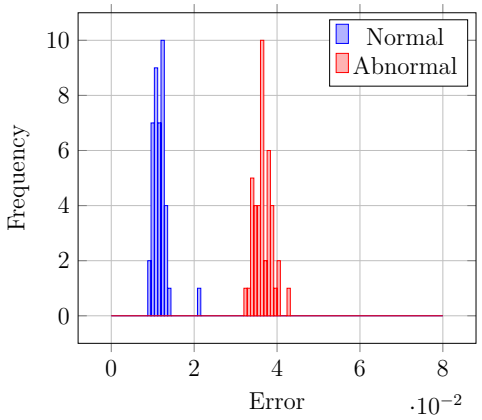


Figure 5.16: Obtained reconstruction error distributions for normal and abnormal test logs respectively after applied to the CNN-AE within cluster 1.

Metric	Score
Accuracy	0.9756
Precision	1.0
Recall	0.9512

Table 5.2: CNN-AE performance on test logs in cluster 1.

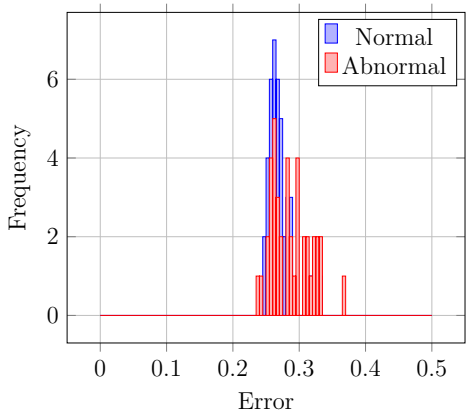


Figure 5.17: Obtained reconstruction error distributions for normal and abnormal test logs respectively after applied to the LSTM-AE within cluster 1.

Metric	Score
Accuracy	0.6098
Precision	0.5672
Recall	0.9268

Table 5.3: LSTM-AE performance on test logs in cluster 1.

6

Discussion

The following chapter discusses the obtained results from this study, along with further improvements and drawn conclusions. In specific, Section 6.1 discusses obtained results from the gear shift type classification and Section 6.2 discusses findings regarding abnormal gear shift prediction. Section 6.3 reflects upon further improvements commenting on the provided data and chosen work process. Lastly, Section 6.4 describes drawn conclusions.

6.1 Gear Shift Type Classification

The results from the HDBSCAN clustering indicate that distinct clusters were formed. Only a small number of logs that were mostly in between clusters were classified as noise by the clustering algorithm. As visualized in Figure 5.2, two larger clusters (2 and 7) were formed along with seven smaller ones. Since there was more training data within the larger clusters, one concern was that the CNNClassifier would classify the logs in larger clusters better than the logs in the smaller clusters. However, since the classifier achieved an accuracy of 98%, this was not the case.

By looking at Figure 5.5, which shows samples from each classified cluster, one can see that cluster five and six have mixed gear shift logs. There are different reasons as to why this occurred. First thought is that the HDBSCAN clustering was not perfectly accurate, resulting in clusters with logs that might not have been correctly classified. Another interpretation is that even though the CNNClassifier achieved a high accuracy it was still not 100% and since the dataset consisted of thousands of logs, even a small percentage could lead to multiple misclassified logs.

One challenge with clustering without having ground truth labels was to evaluate how well the clusters were separated. The obtained silhouette score of 0.513 indicate that the clusters were well separated but to blindly depend on this score is not reliable. Although a score closer to 1 would have been preferred, the classification of different shifts aligns with the thesis scope and thus obtained results are satisfactory.

6.2 Abnormal Gear Shift Prediction

Regarding abnormal gear shift prediction, several observations linked to the training and evaluation results have been noted. A comparison of the performance scores in Tables 5.1, 5.2 and 5.3, show that the CNN architecture achieved the best results with values above 95%. However, the sorted gear shift logs in Figures 5.8 and 5.14 indicate that the other architectures might have sorted the logs better. By comparing Figure 5.11 with Figures A.4, A.13 and A.22, it seems like the four logs with lowest reconstruction errors in Figure 5.11 belong to cluster 4 which makes the obtained results from the CNN-AE unreliable. This is believed to be a consequence of that the CNN-AE reconstructs gear shift logs with much lower error compared to the other architectures and the difference in errors between normal and abnormal logs is not as widespread. However, since the given data was completely unlabeled, the probability of training the autoencoder on clusters without mixed gear shift types was low. This makes the training skewed, since there is a risk that abnormal gear shifts are interchanged with normal gear shifts. Also, reconstructed data is compared to abnormal data contradicting the intended way of training the autoencoders for anomaly detection.

Furthermore, for evaluation of the autoencoders, a self-generated test set was constructed where noise was added in order to label test logs as abnormal. This approach is considered to have at least two challenges. Firstly, the addition of Gaussian noise to part of the test data, is likely correlated to the shape of the logs. Since autoencoders are able to reconstruct different types of shapes, uncorrelated noise would have been preferable although this is outside the thesis scope. Secondly, the part of the test data where no noise was added might still have contained abnormal gear shift logs. Even though these were assumed to be less than 10% of the remaining logs, they could have negatively affected the results.

6.3 Further Improvements

Provided Data

As previously mentioned, the available data in this study was unlabeled which caused some difficulties in obtaining reliable results. The autoencoders could have taken advantage of training on ground truth labels to depict example logs that were classified as belonging to different gear shift types and to know in advance which logs are classified as normal or not. Another possible improvement in this sense is therefore to create a labeled test data set that could evaluate the classifier and autoencoder performance through the use of ground truth. For example, experts in the field and experienced truck drivers could give a review and annotate gear shifts that they believe are abnormal, although this also entails risks in the form of the data becoming subjective.

In addition to the unlabeled data, it is likely that there are further improvements regarding the data size. The amount of available data processed by the model was

based on 5245 gear shift logs and this has been considered sufficient to form an opinion that it is possible to implement the model as a tool for the detection of abnormal data within truck gear shifts. Also. since density-based clustering is dependent on the number of logs, the clustering is believed to improve with more training data. Thus, as a consequence the autoencoders could potentially be able to detect abnormal gear shifts more accurately.

Research Methodology

The workflow, illustrated in Figure 4.1, was overall effective from start to finish where each step of the process contributed to the final outcome. Since the related work studied in section 1.2 were based on similar methods and were deemed successful, the chosen work process was expected to work. However, certain steps in the process pose difficulties and could be further improved.

One important step was to transform the multivariate data to a univariate format by flattening the data. On one hand, flattening the data makes multivariate analysis much easier, but on the other hand, it can pose unwanted challenges for both classification and fault detection. One notable issue arised when two logs were identical except for one signal, like cluster 6 in Figure 5.5. In such cases, they were grouped into the same cluster but in reality they could be two different gear shift types. Another concern is whether the performance of the autoencoder is affected by the flattened data. Since the autoencoder attempts to reconstruct the entire flattened data, which contains all sensors, there is a potential risk that abnormal logs would not be detected. If only one of the signals is abnormal, the reconstruction error could be lower than if multiple sensors are abnormal. Consequently, the log with only one abnormal signal could have lower reconstruction error than what the threshold is set to. A future improvement could be to capture abnormalities for both univariate and multivariate data simultaneously.

One of the limitations in the study was to maintain the simplicity of the autoencoders to fully understand how the model processes the data. For this reason, only a few layers in the autoencoders were used. By comparing the architectures, Tables 5.1, 5.2 and 5.3 showed that the dense and CNN autoencoders performed better than the LSTM-AE. Furthermore, Figure 5.17 shows that the LSTM-AE had overlapping reconstruction error distributions meaning that normal and abnormal test logs could not be separated. The reason for this is not fully clear, but further fine-tuning of the LSTM architecture is believed to improve its performance.

6.4 Conclusions

The aim of this study was to develop a machine learning model that would manage to classify unlabeled gear shifts and detect abnormal sensor data within heavy-duty trucks. Obtained results are regarded as promising and show potential in using machine learning as a future tool for aiding engineers analyze field test data and detect abnormalities. The CNNClassifier achieved an overall accuracy of about 98%, indi-

cating it as a feasible choice for classifying diverse gear shift types. Furthermore, the CNN-AE architecture outperformed the other autoencoders showing its potential in predicting abnormal gear shifts with accuracy, precision and recall scores above 95%.

However, it is concluded that further improvements are needed in order to get more distinct clusters and more accurate predictions without mixed gear shift types. As mentioned in the discussion, either a better way of adding noise that is uncorrelated to the sensor data or creating a labeled test data set would make the obtained results more reliable.

Bibliography

[1] Volvo Trucks. Volvo trucks' i-shift transmission technology is still a breakthrough innovation after 20 years. https://www.volvotrucks.com/en-en/news-stories/press-releases/2021/mar/volvo-trucks-i-shift-transmission-technology.html, 2021. [Accessed: 2024-05-20].

[2] Volvo Lastvagnar. Upptäck i-shift – den automatiserade växellådan. https://www.volvotrucks.se/sv-se/trucks/features/i-shift.html, May 2024. [Accessed: 2024-05-20].

[3] Scott Hulbert, Calahan Mollan, and Vijitashwa Pandey. Fault diagnosis and prediction in automotive systems with real-time data using machine learning. Technical report, SAE Technical Paper, https://doi.org/10.4271/2022-01-0217, 2022.

[4] Iron Tessaro, Viviana Cocco Mariani, and Leandro dos Santos Coelho. Machine learning models applied to predictive maintenance in automotive engine components. In *Proceedings*, volume 64, page 26. MDPI, 2020.

[5] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):1–58, 2009.

[6] Narjes Davari, Sepideh Pashami, Bruno Veloso, Sławomir Nowaczyk, Yuantao Fan, Pedro Mota Pereira, Rita P Ribeiro, and João Gama. A fault detection framework based on lstm autoencoder: A case study for volvo bus data set. In *International Symposium on Intelligent Data Analysis*, pages 39–52. Springer, 2022.

[7] Daniel Stenekap. *Classification of Gear-shift data using machine learning*. Master's thesis, Mälardalens högskola, https://urn.kb.se/resolve?urn=urn:nbn:se:mdh:diva-53445, 2021.

[8] Andreas Nordberg. *Evaluation of Neural Networks for Predictive Maintenance: A Volvo Penta Study*. Master's thesis, Linköpings universitet, https://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-176390, 2021.

[9] Volvo Group. Climate. https://www.volvogroup.com/en/sustainable-transportation/responsible-business/climate.html, 2023. [Accessed: 2024-05-20].

[10] Volvo Trucks. How can the trucking industry be a part of the circular economy? https://www.volvotrucks.com/en-en/news-stories/insights/articles/2019/oct/how-can-the-trucking-industry-be-a-part-of-the-circular-economy.html, 2019. [Accessed: 2024-05-20].

[11] H. P. Vinutha, B. Poornima, and B. M. Sagar. Detection of outliers using interquartile range technique from intrusion dataset. In Suresh Chandra Satapathy, Joao Manuel R.S. Tavares, Vikrant Bhateja, and J. R. Mohanty, editors, *Information and Decision Sciences*, pages 511–518, Singapore, 2018. Springer Singapore.

[12] Stephen Allwright. Mse vs mae, which is the better regression metric? https://stephenallwright.com/mse-vs-mae/, 2022. [Accessed: 2024-05-20].

[13] Cort J Willmott and Kenji Matsuura. Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. *Climate research*, 30(1):79–82, 2005.

[14] Timothy O Hodson. Root mean square error (rmse) or mean absolute error (mae): When to use them or not. *Geoscientific Model Development Discussions*, 2022:1–10, 2022.

[15] Simone Santini and Ramesh Jain. Similarity measures. *IEEE Transactions on pattern analysis and machine Intelligence*, 21(9):871–883, 1999.

[16] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[17] A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):210–229, 1959.

[18] Julianna Delua. Supervised vs. unsupervised learning: What's the difference? https://www.ibm.com/blog/supervised-vs-unsupervised-learning/, 2021. [Accessed 24-04-2024].

[19] Warren S. Mcculloch. The brain computing machine. *Electrical Engineering*, 68(6):492–497, 1949.

[20] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5:115–133, 1943.

[21] Bernhard Mehlig. *Machine Learning with Neural Networks: An Introduction for Scientists and Engineers*. Cambridge University Press, October 2021.

[22] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[23] Ahmed Gad. A comprehensive guide to the backpropagation algorithm in neural networks. https://neptune.ai/blog/backpropagation-algorithm-in-neural-networks-guide, 2023. [Accessed 11-01-2024].

[24] Cole Stryker Dave Bergmann. What is an autoencoder? https://www.ibm.com/topics/autoencoder, 2023. [Accessed 22-02-2024].

[25] Serkan Kiranyaz, Onur Avci, Osama Abdeljaber, Turker Ince, Moncef Gabbouj, and Daniel J. Inman. 1d convolutional neural networks and applications: A survey. *Mechanical Systems and Signal Processing*, 151:107398, 2021.

[26] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[27] Stephen M. Walker II. What are accuracy, precision, recall, and f1 score? https://klu.ai/glossary/accuracy-precision-recall-f1, April 2023. [Accessed: 2024-05-20].

[28] Hans-Peter Kriegel, Peer Kröger, Jörg Sander, and Arthur Zimek. Density-based clustering. *Wiley interdisciplinary reviews: data mining and knowledge discovery*, 1(3):231–240, 2011.

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

[30] Mihael Ankerst, Markus M Breunig, Hans-Peter Kriegel, and Jörg Sander. Optics: Ordering points to identify the clustering structure. *ACM Sigmod record*, 28(2):49–60, 1999.

[31] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[32] CAN in Automation. History of can technology. https://www.can-cia.org/can-knowledge/history-of-the-can-technology, 2024. [Accessed 22-02-2024].

[33] Canalyzer – ecu network analysis | vector. https://www.vector.com/int/en/products/products-a-z/software/canalyzer/. [Accessed 20-05-2024].

[34] MN Noor, AS Yahaya, Nor Azam Ramli, and AM Mustafa Al Bakri. Filling missing data using interpolation methods: Study on the effect of fitting distribution. *Key Engineering Materials*, 594:889–895, 2014.

[35] Markus Löning, Anthony Bagnall, Sajaysurya Ganesh, Viktor Kazakov, Jason Lines, and Franz J Király. sktime: A unified interface for machine learning with time series. *arXiv preprint arXiv:1909.07872*, 2019.

A
Appendix 1

A.0.1 Dense Autoencoder Results

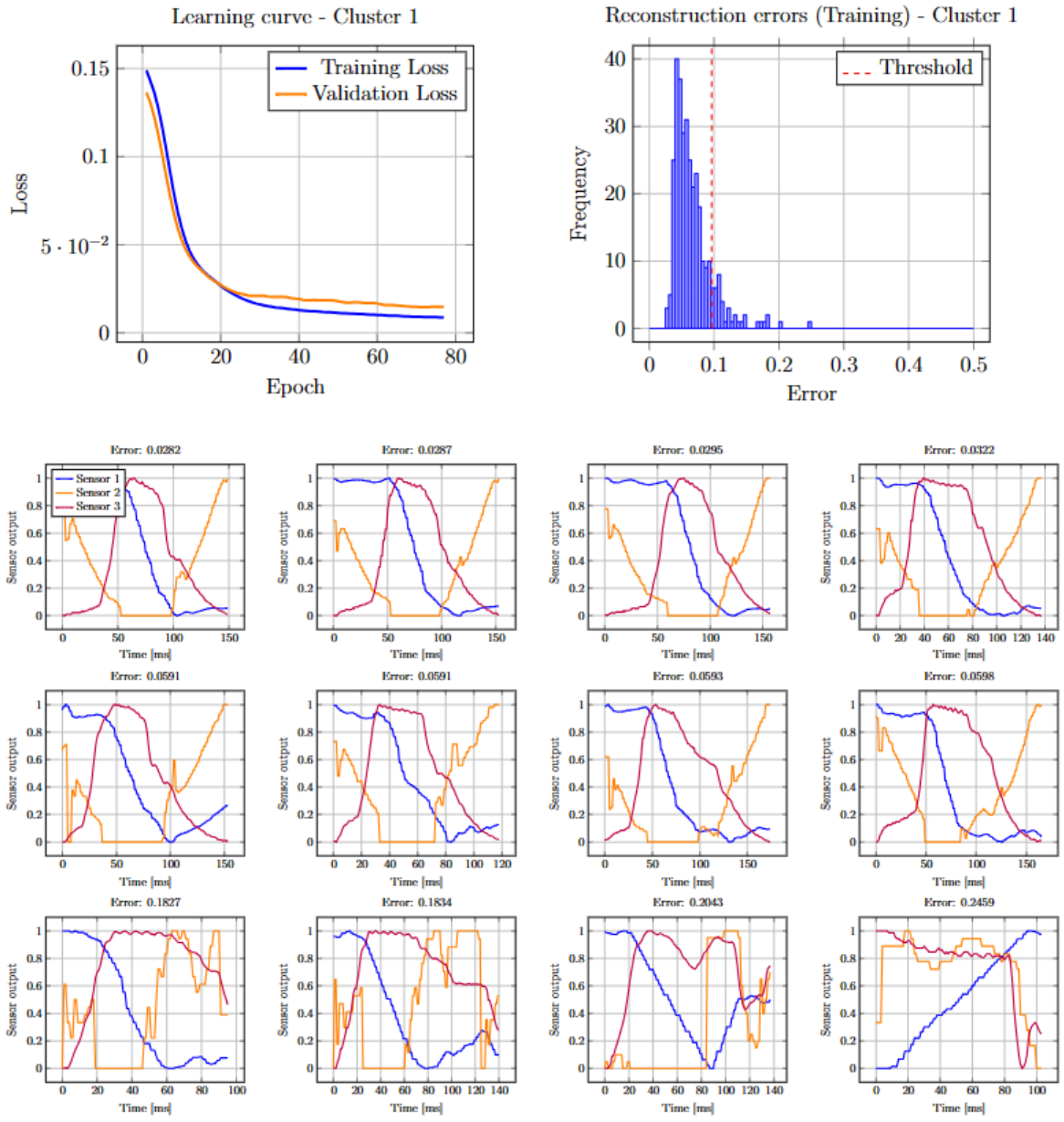


Figure A.1: Training results from the dense autoencoder when applied to gear shift logs within cluster 1.

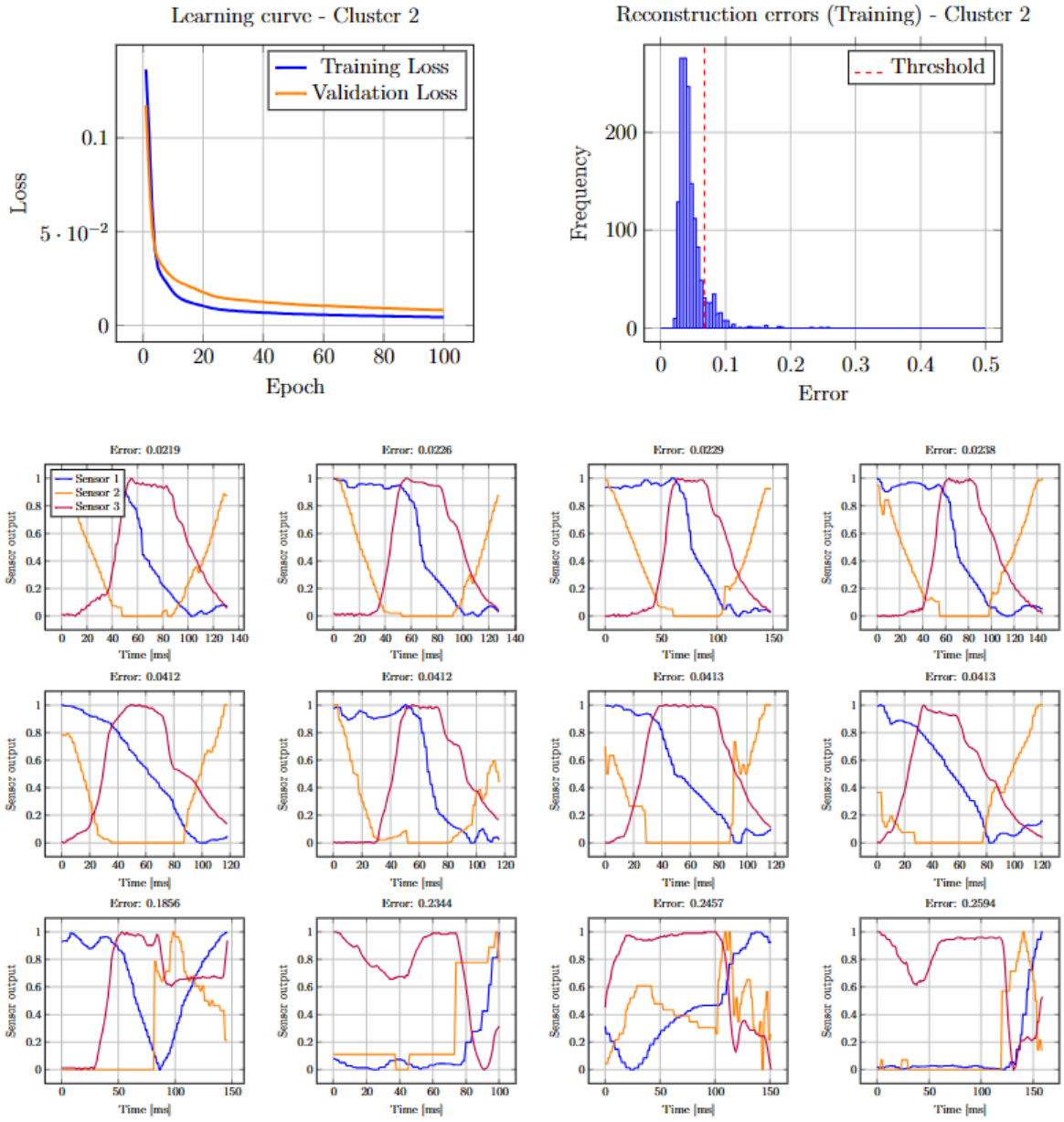


Figure A.2: Training results from the dense autoencoder when applied to gear shift logs within cluster 2.

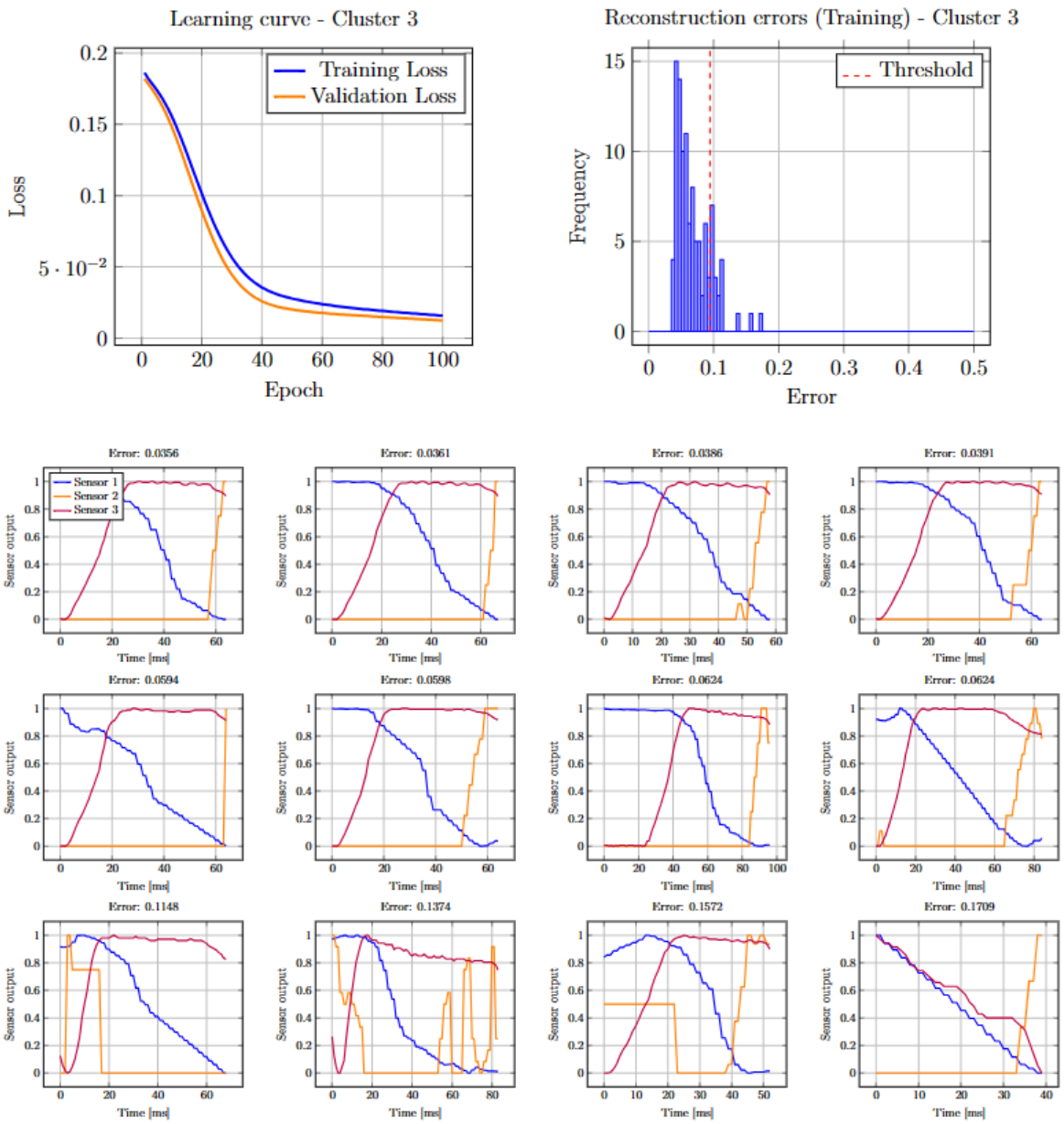


Figure A.3: Training results from the dense autoencoder when applied to gear shift logs within cluster 3.

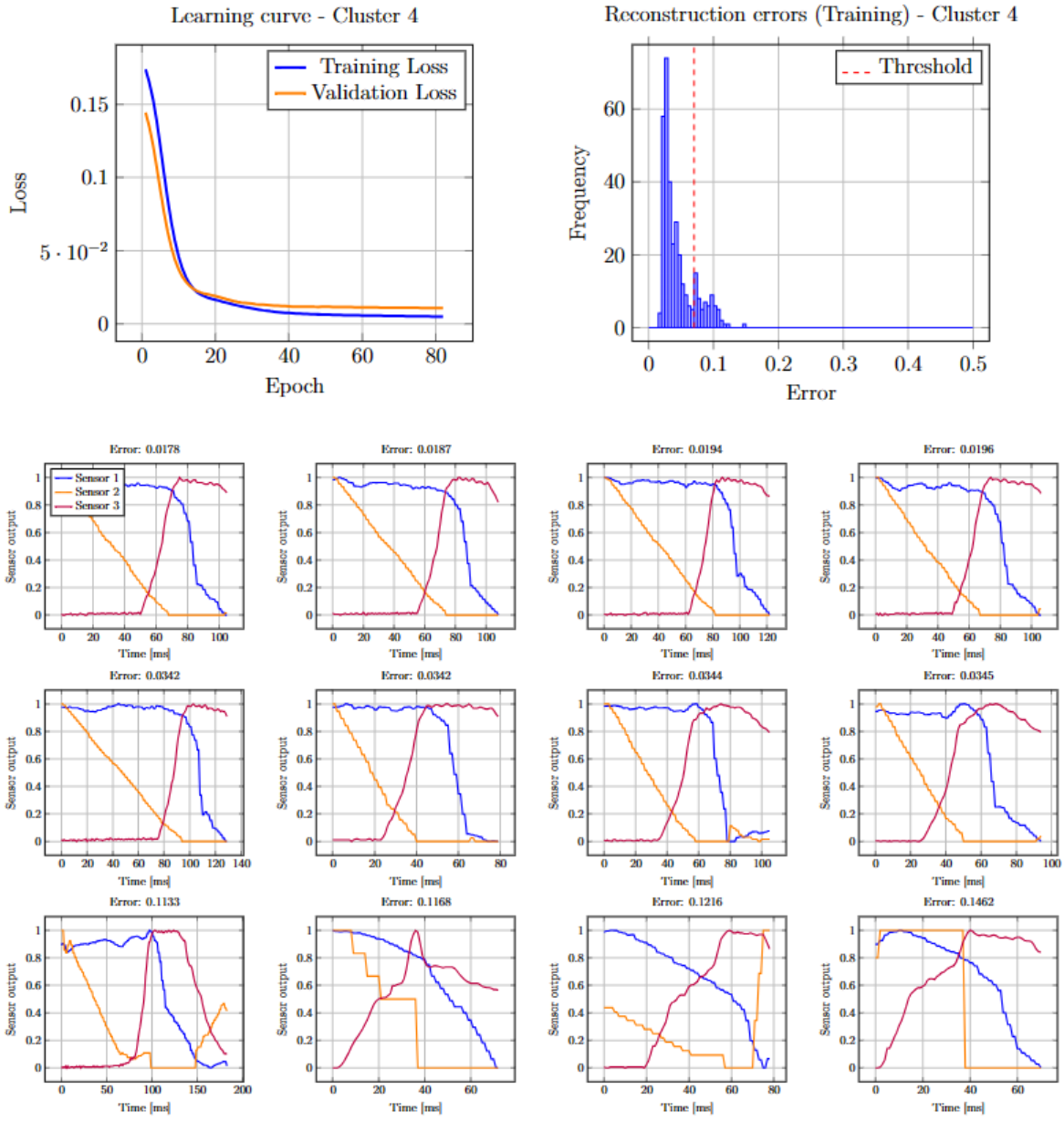


Figure A.4: Training results from the dense autoencoder when applied to gear shift logs within cluster 4.

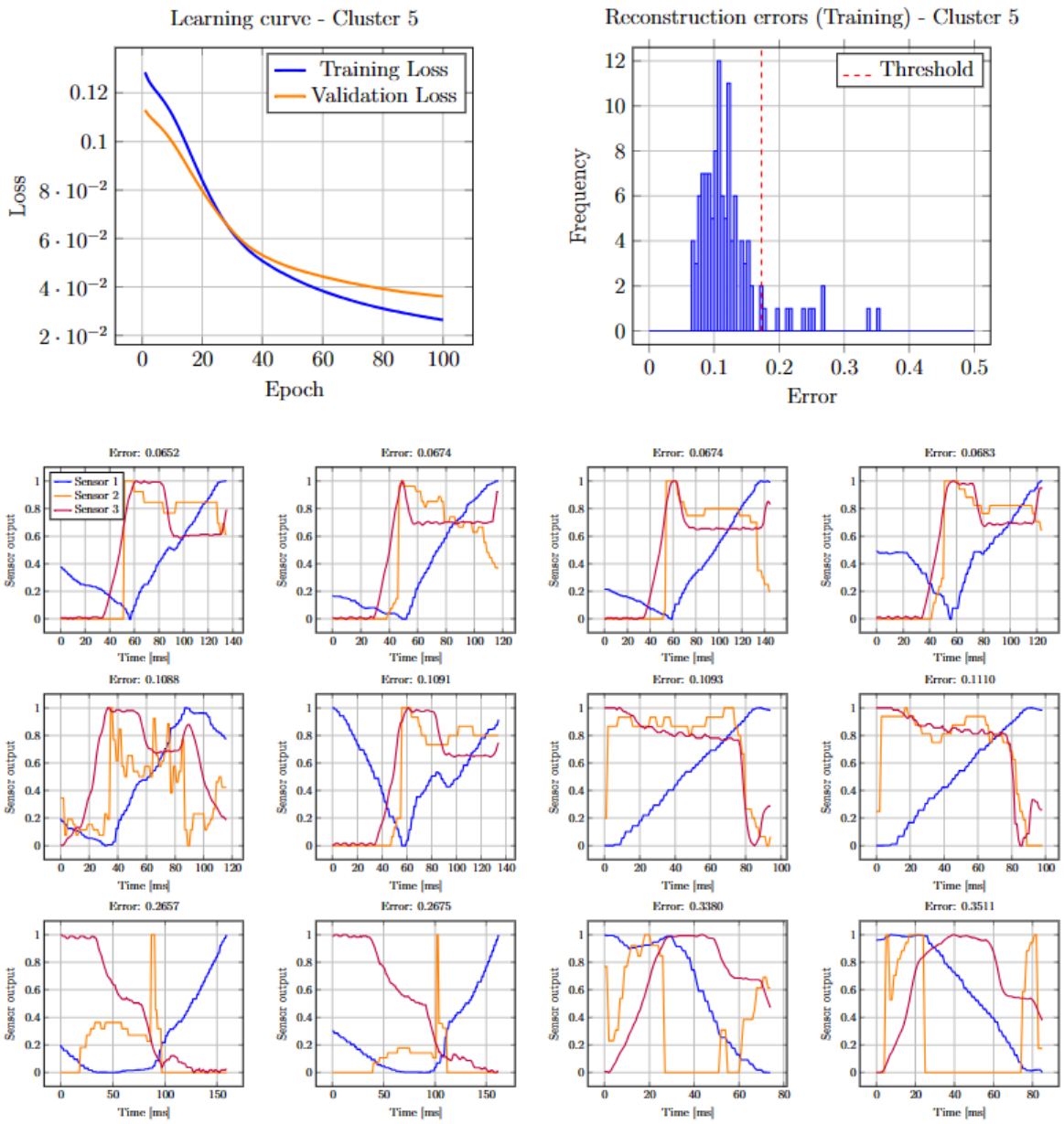


Figure A.5: Training results from the dense autoencoder when applied to gear shift logs within cluster 5.

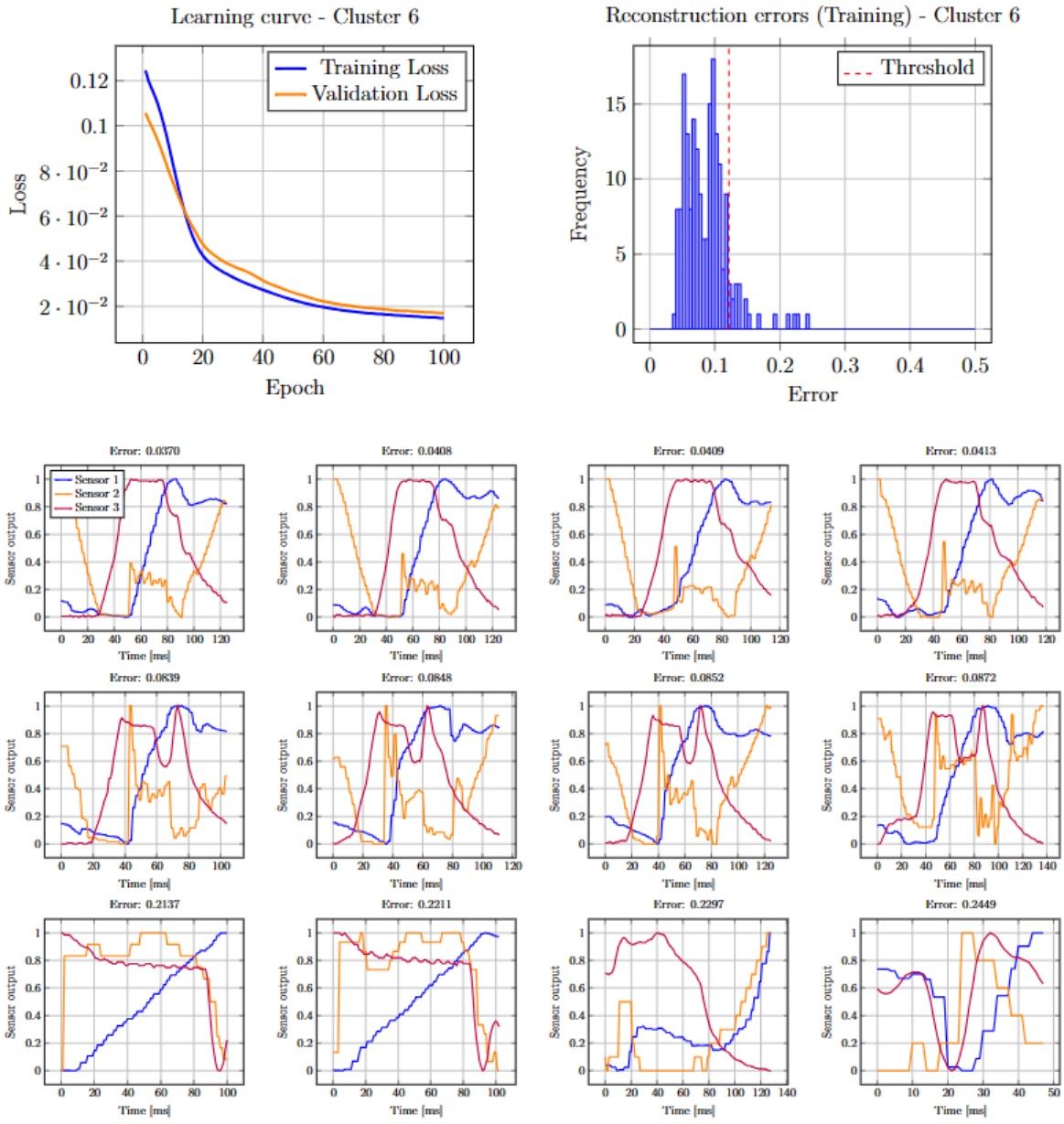


Figure A.6: Training results from the dense autoencoder when applied to gear shift logs within cluster 6.

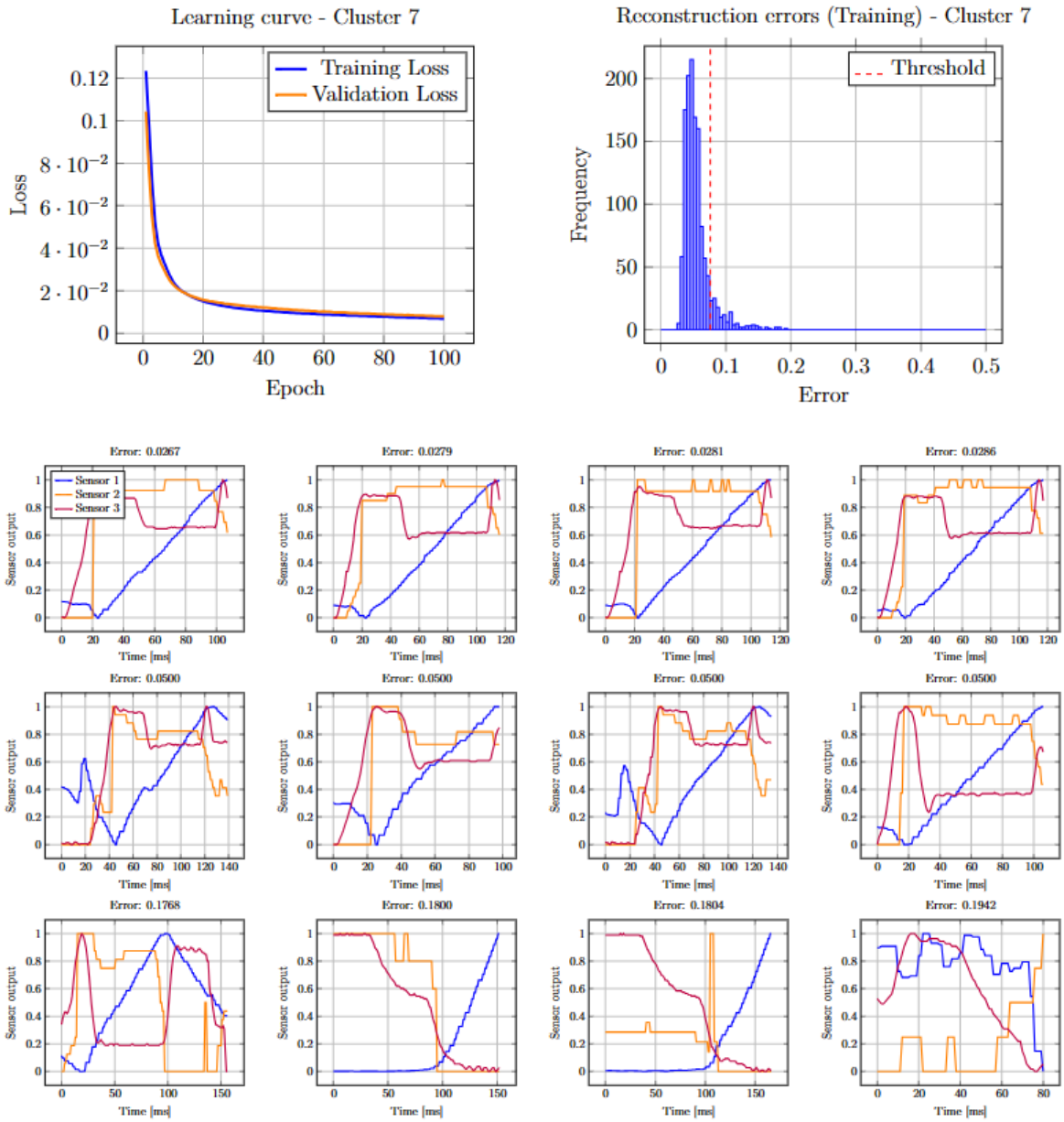


Figure A.7: Training results from the dense autoencoder when applied to gear shift logs within cluster 7.

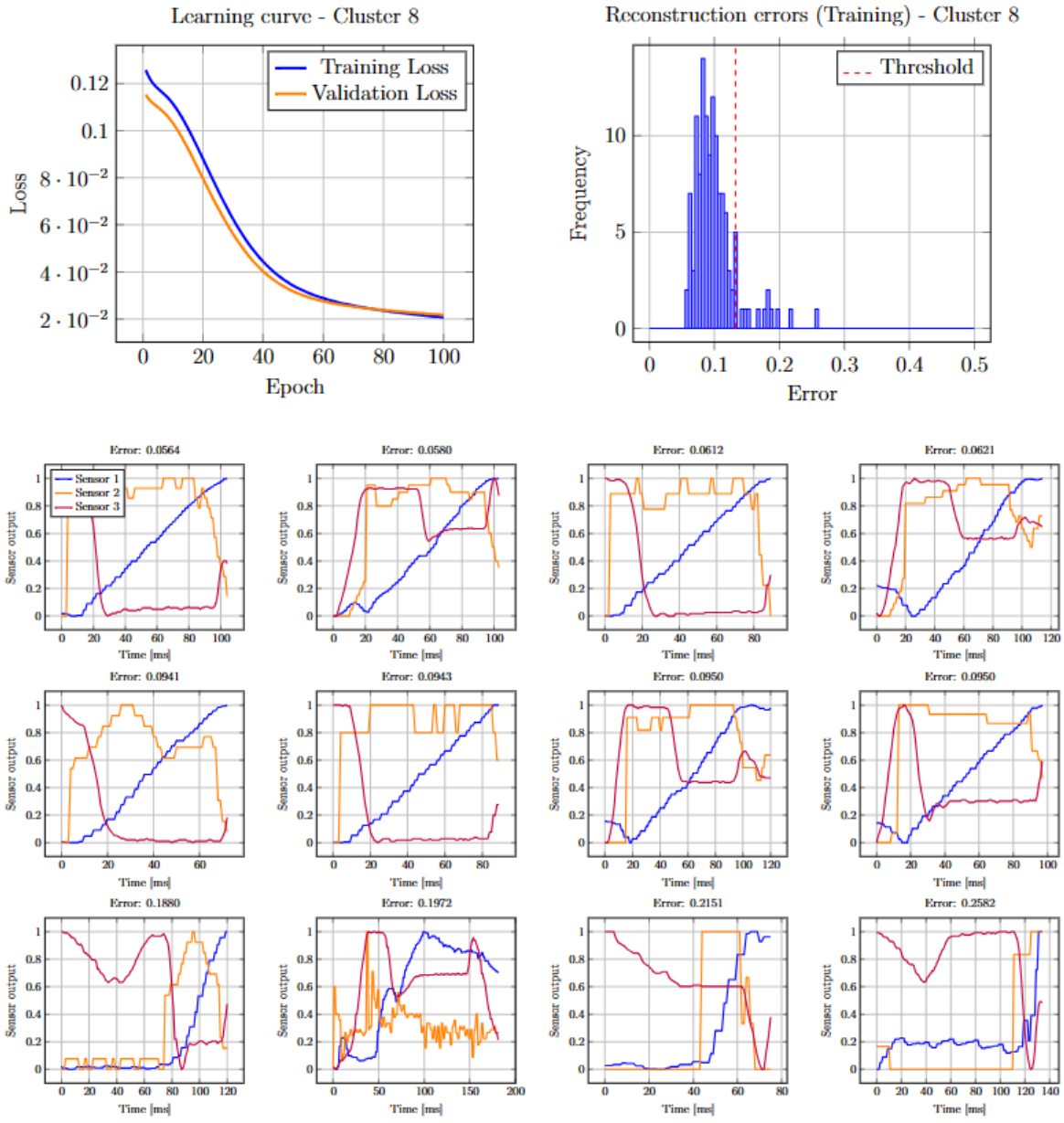


Figure A.8: Training results from the dense autoencoder when applied to gear shift logs within cluster 8.

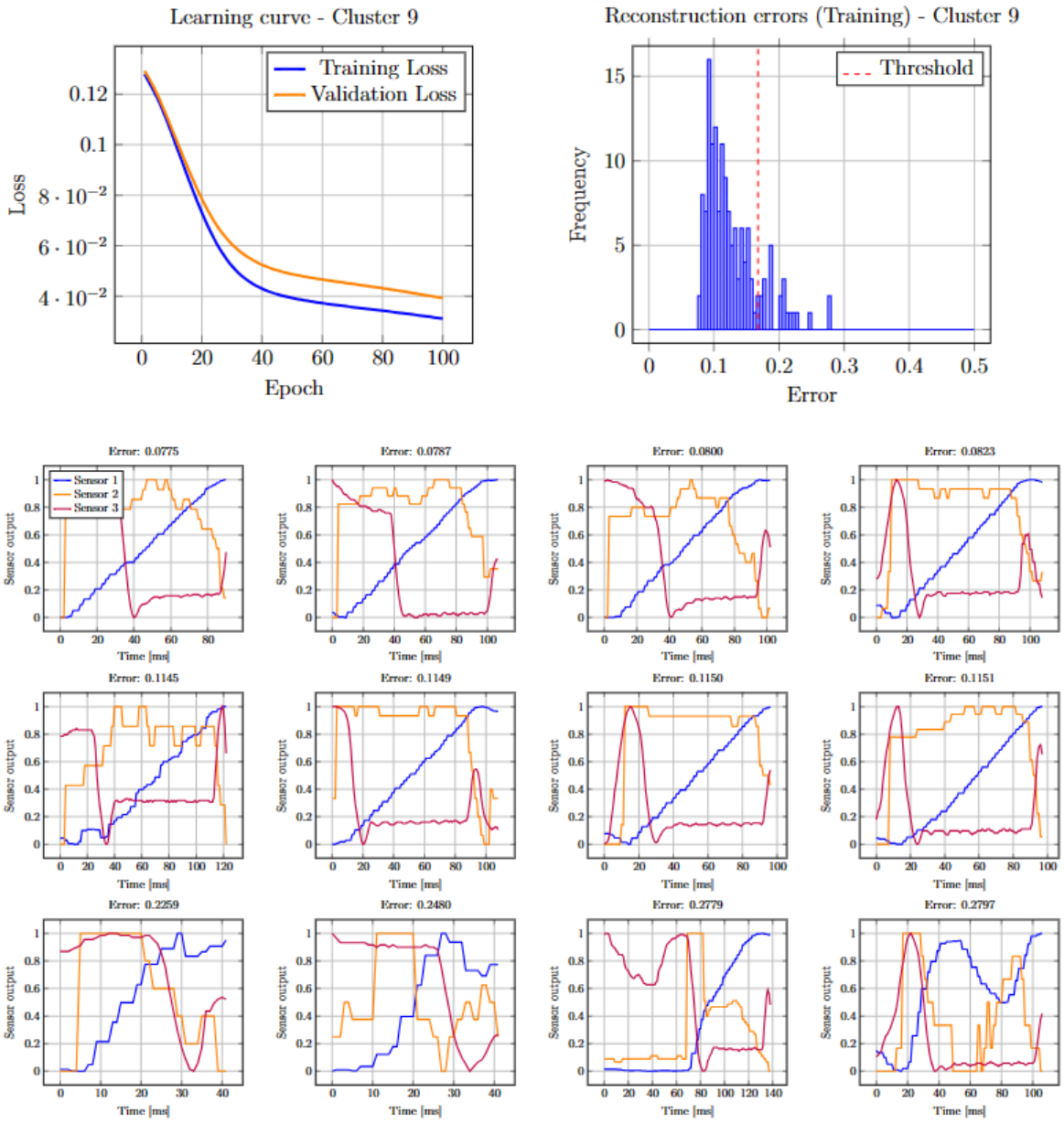


Figure A.9: Training results from the dense autoencoder when applied to gear shift logs within cluster 9.

A.0.2 CNN Autoencoder Results

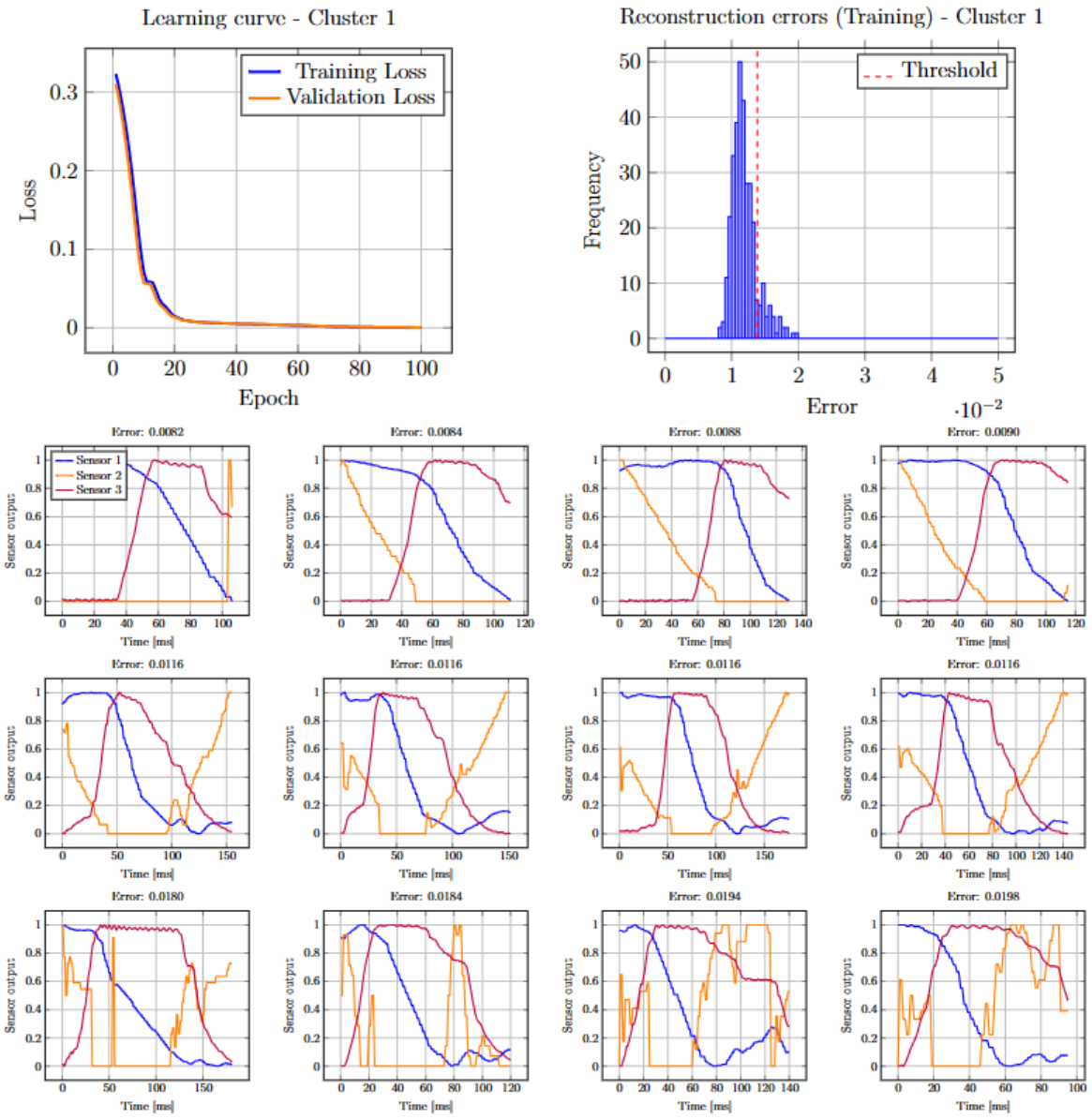


Figure A.10: Training results from the CNN autoencoder when applied to gear shift logs within cluster 1.

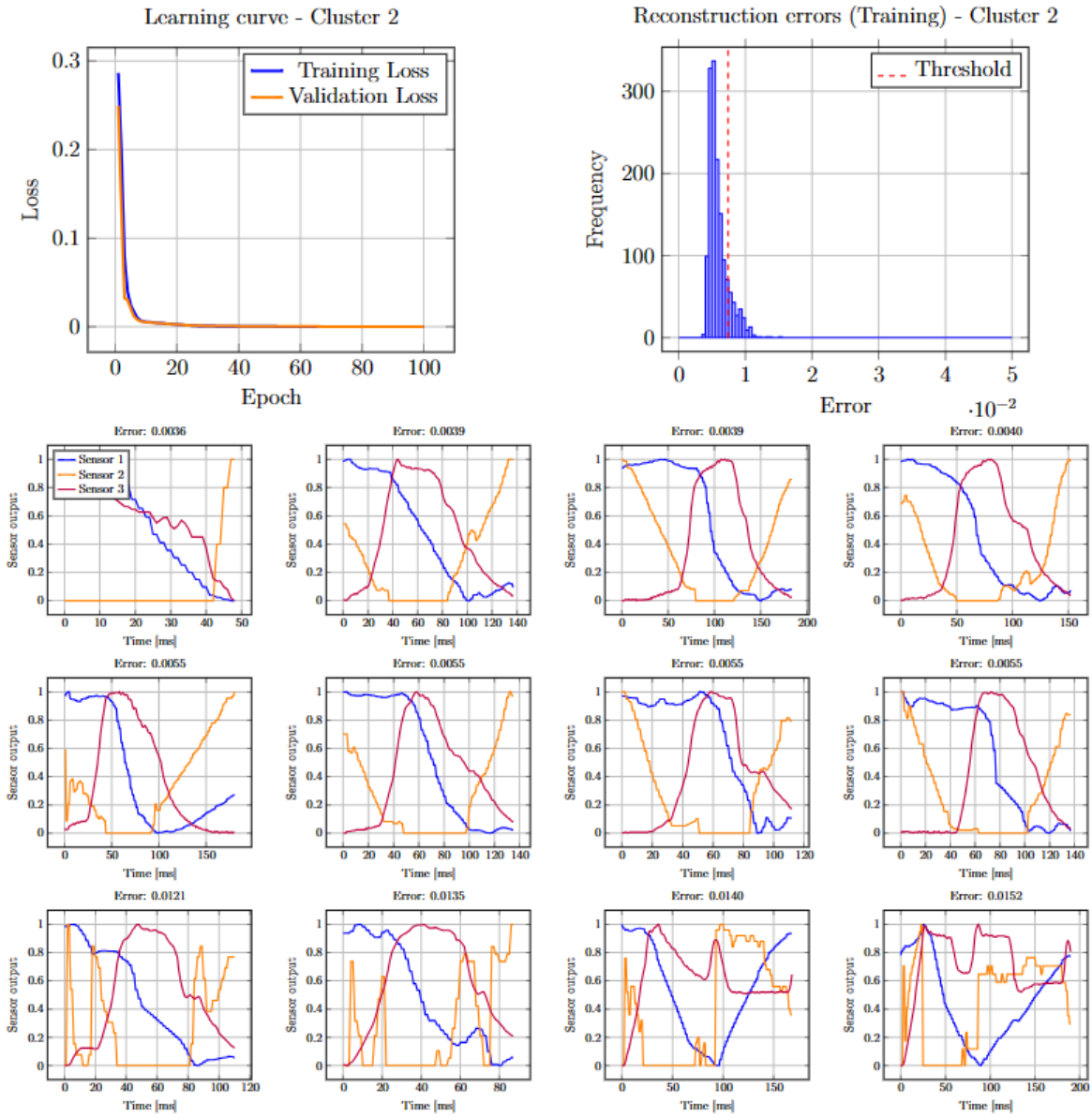


Figure A.11: Training results from the CNN autoencoder when applied to gear shift logs within cluster 2.

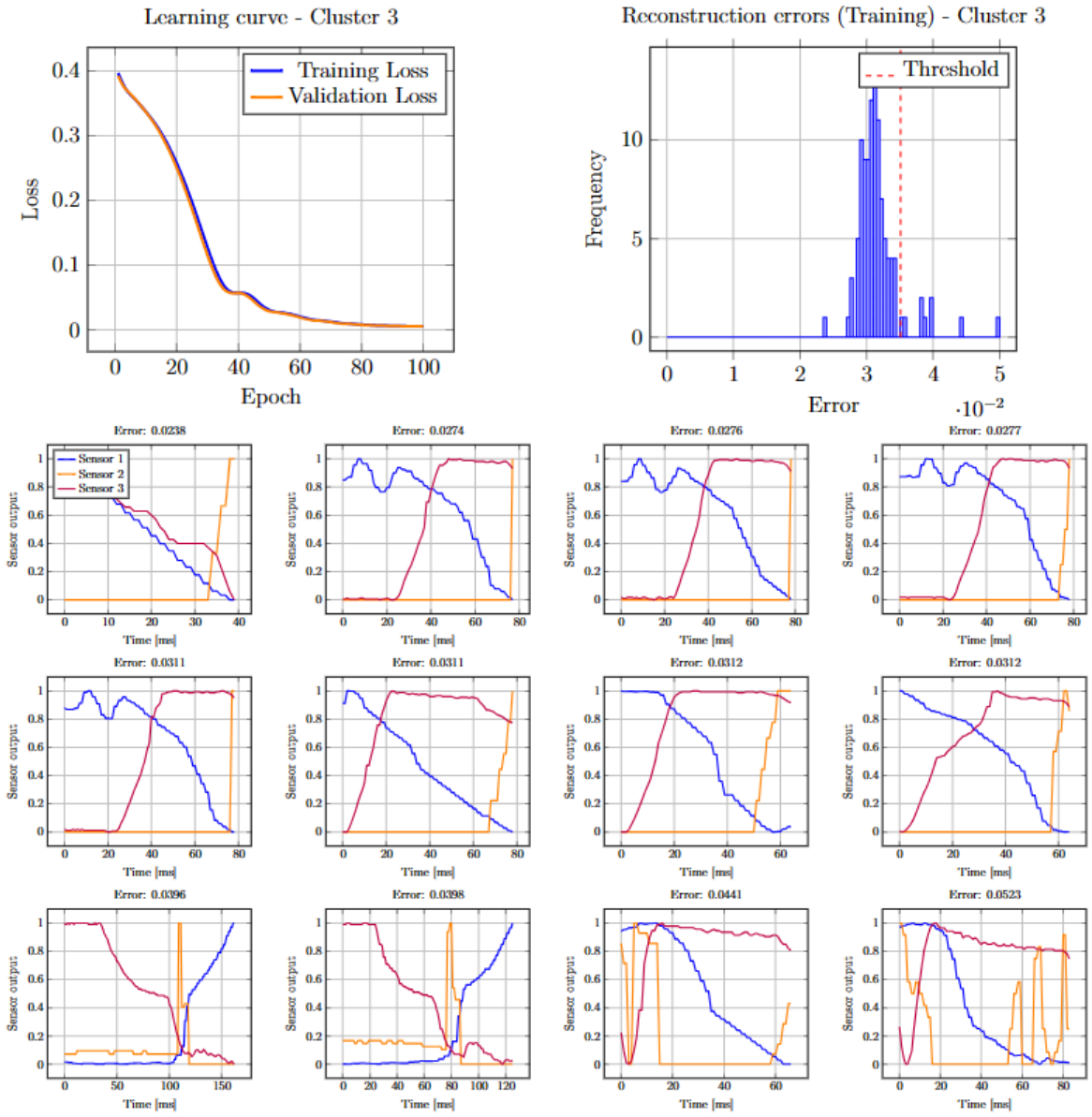


Figure A.12: Training results from the CNN autoencoder when applied to gear shift logs within cluster 3.

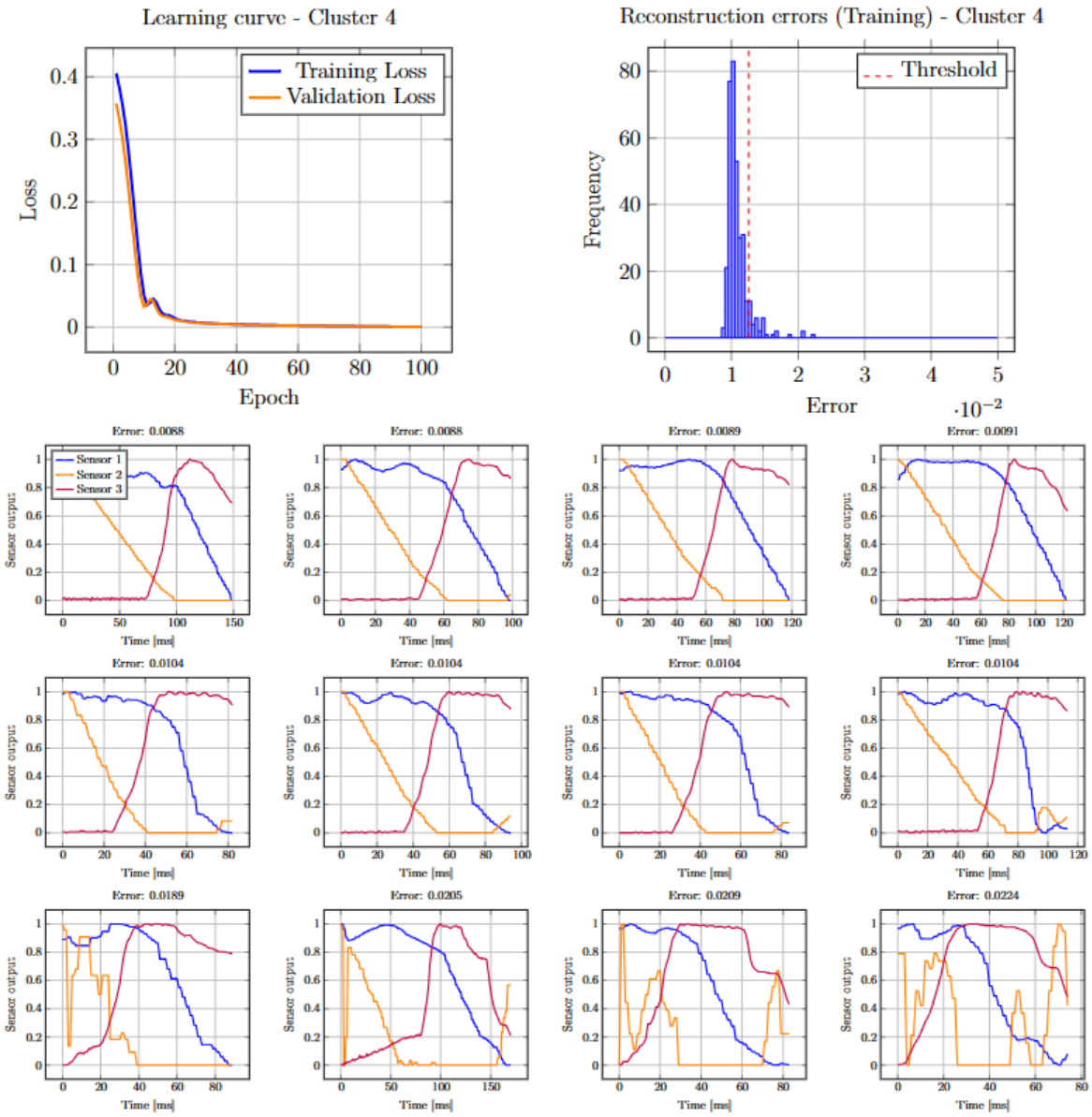


Figure A.13: Training results from the CNN autoencoder when applied to gear shift logs within cluster 4.

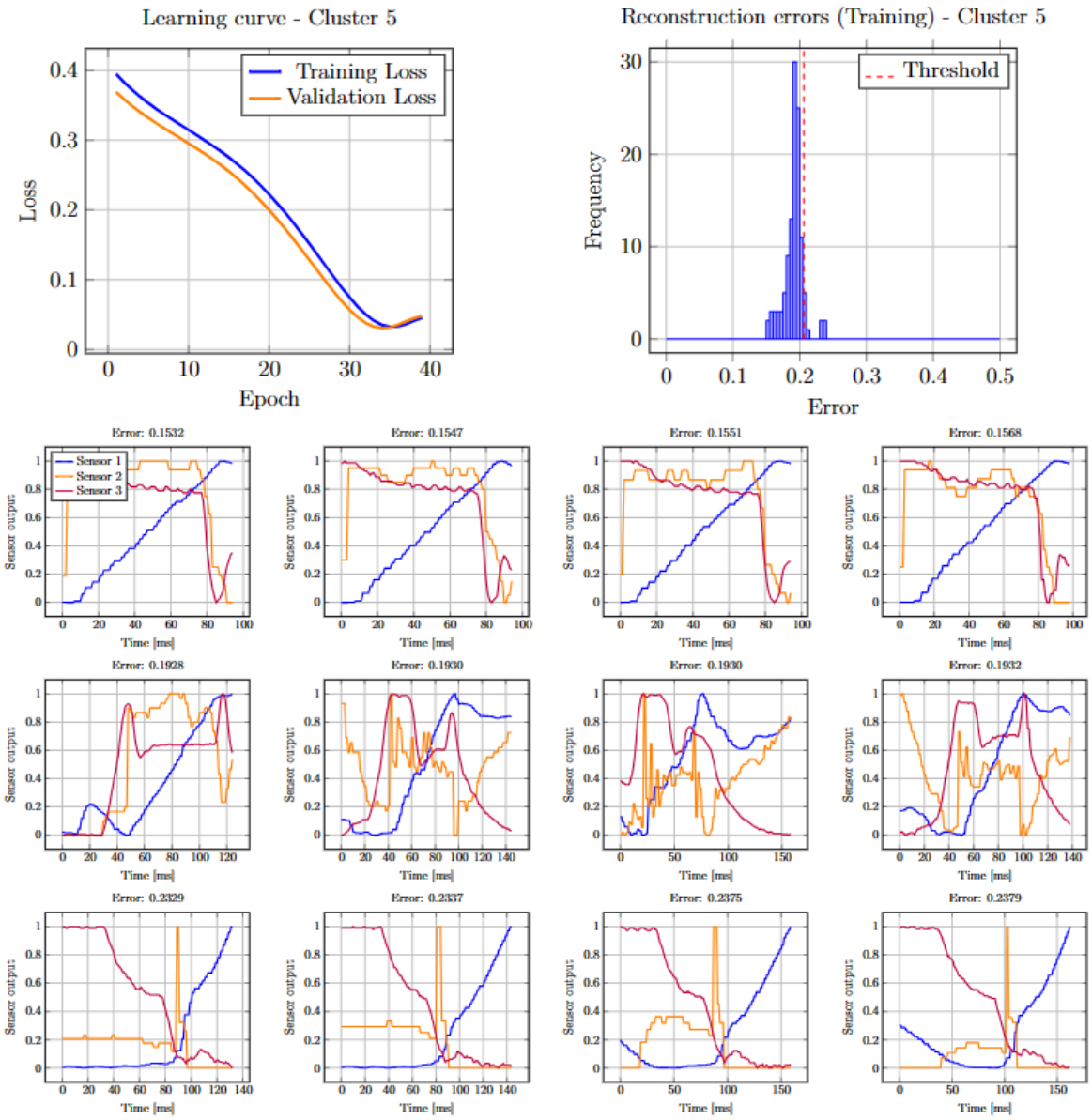


Figure A.14: Training results from the CNN autoencoder when applied to gear shift logs within cluster 5.

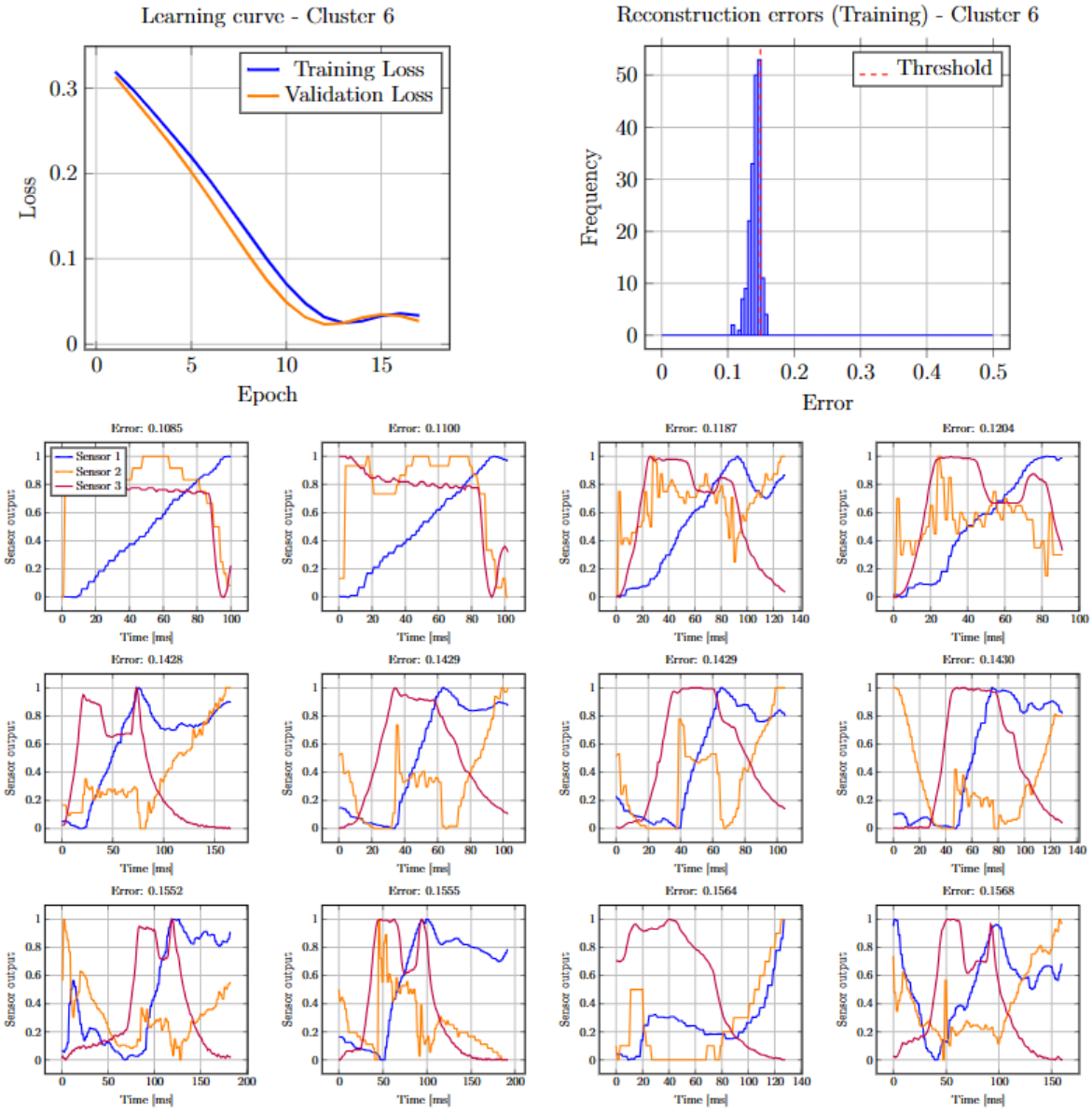


Figure A.15: Training results from the CNN autoencoder when applied to gear shift logs within cluster 6.

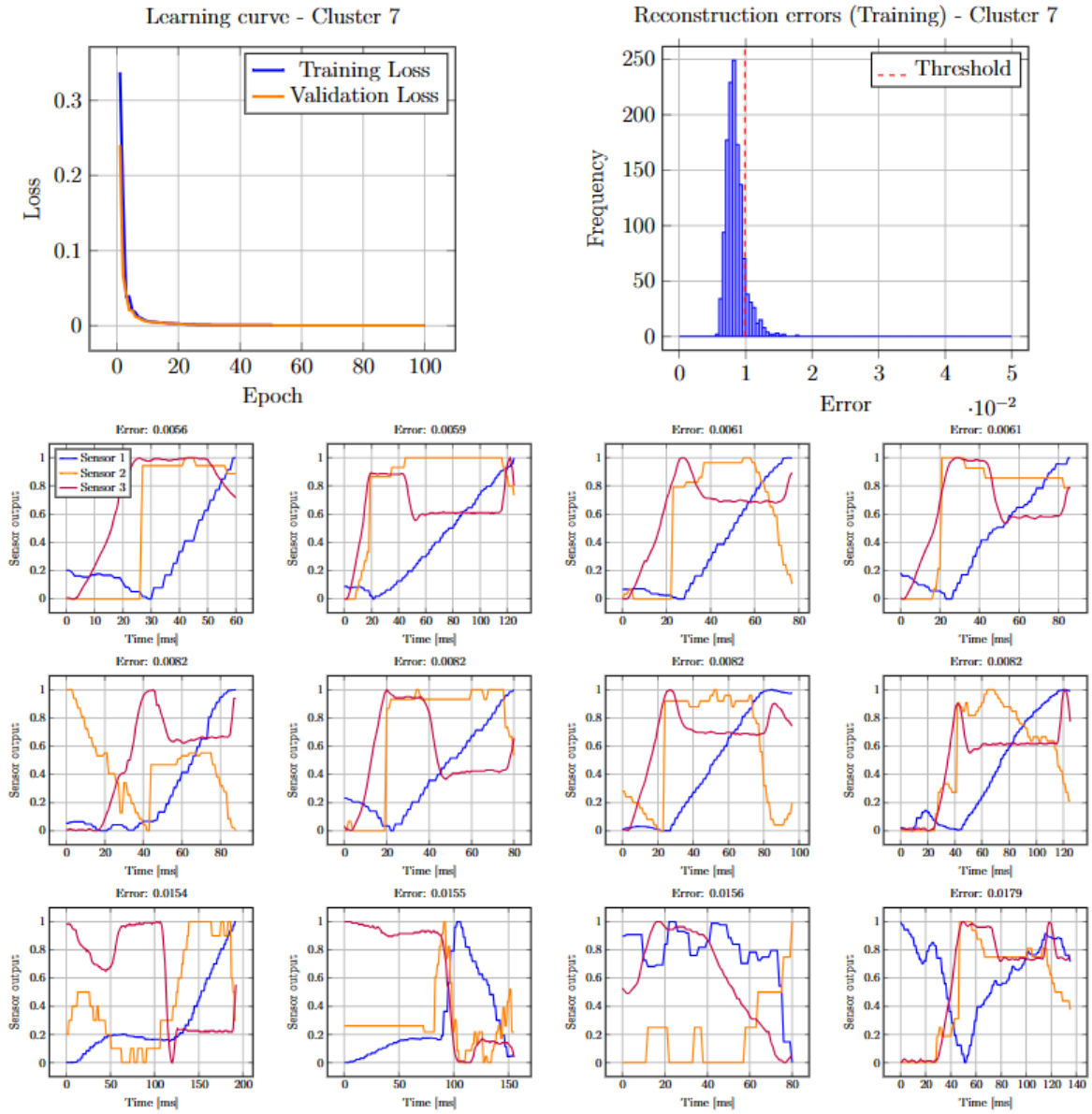


Figure A.16: Training results from the CNN autoencoder when applied to gear shift logs within cluster 7.

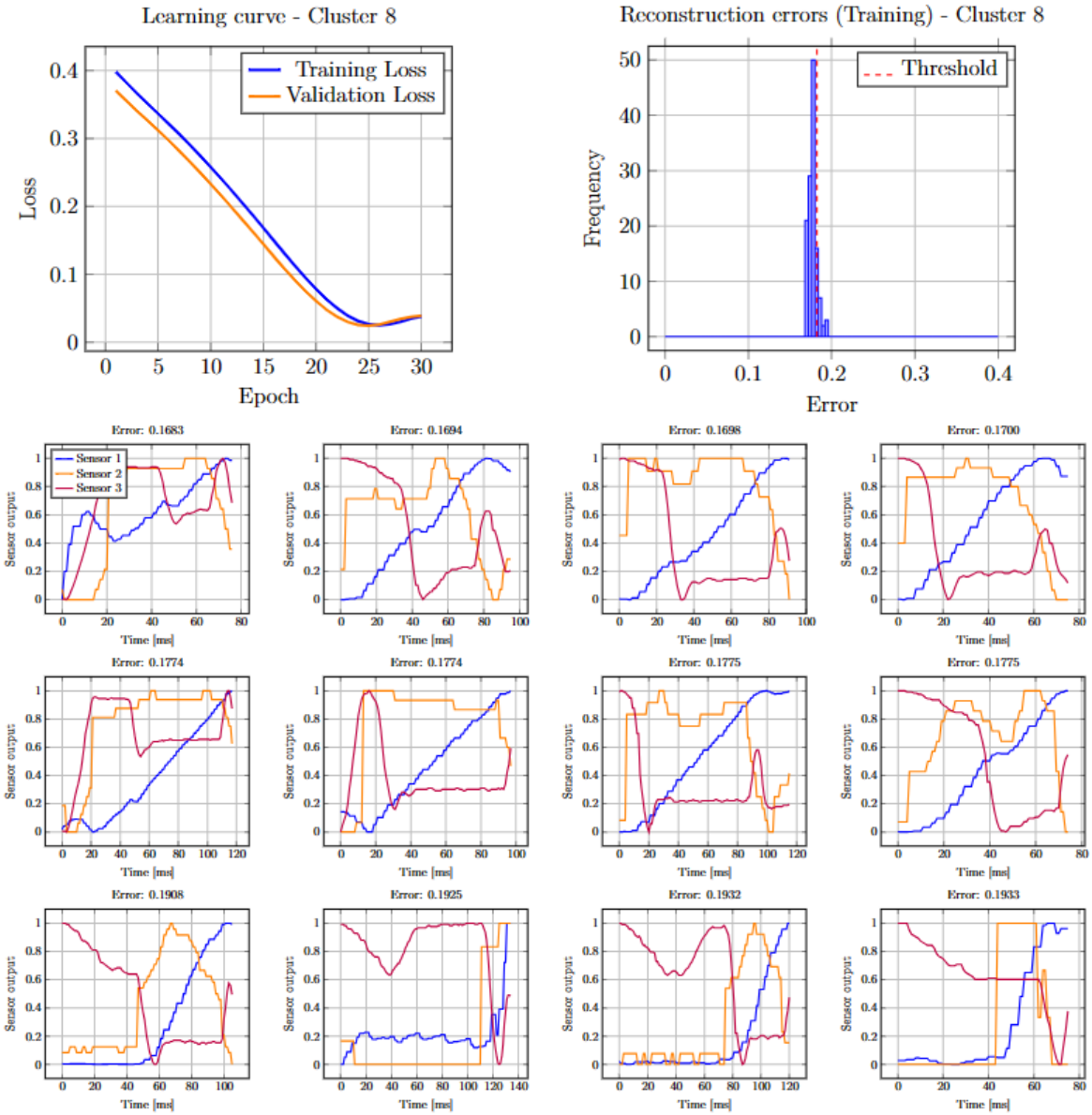


Figure A.17: Training results from the CNN autoencoder when applied to gear shift logs within cluster 8.

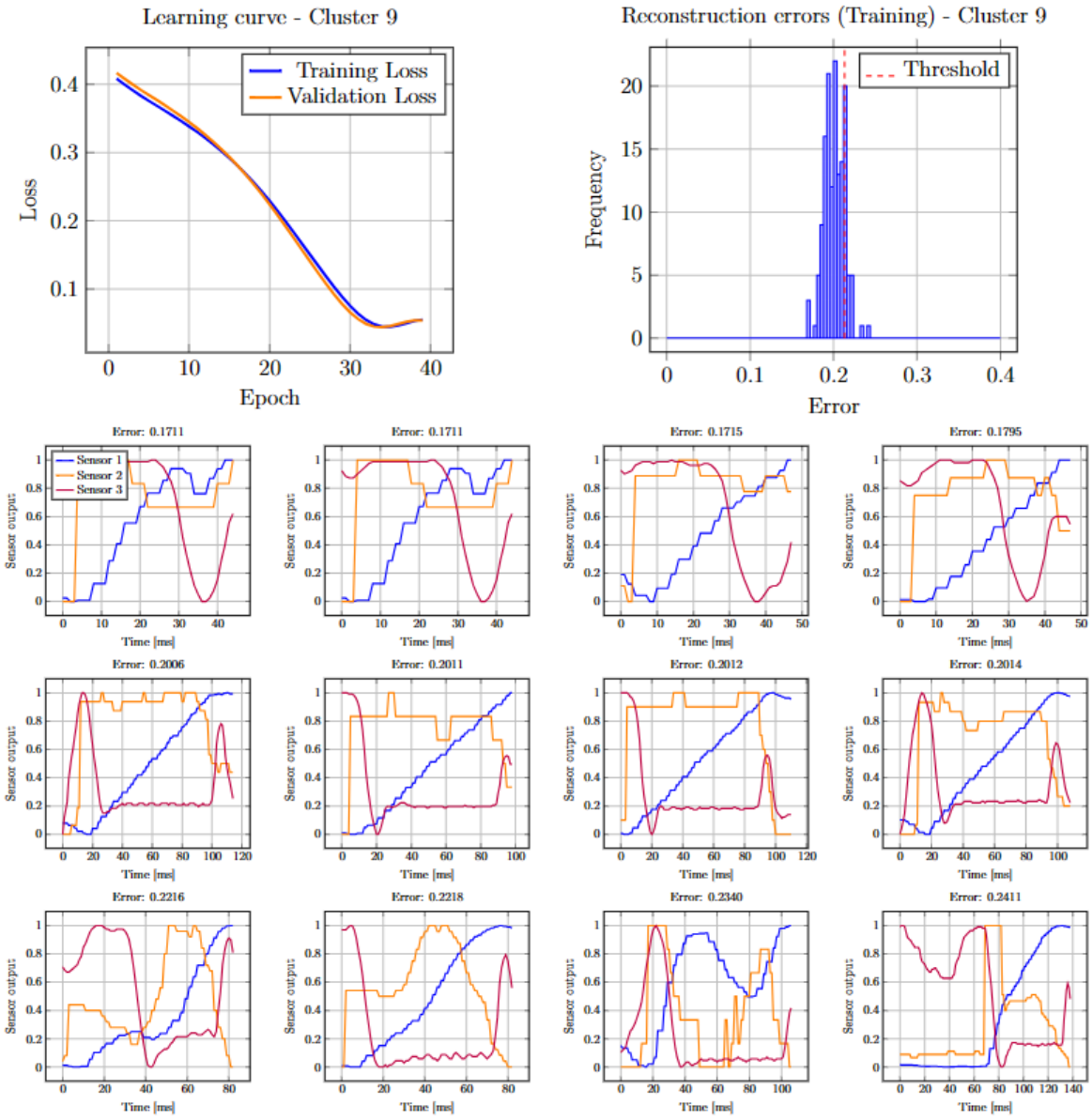


Figure A.18: Training results from the CNN autoencoder when applied to gear shift logs within cluster 9.

A.0.3 LSTM Autoencoder Results

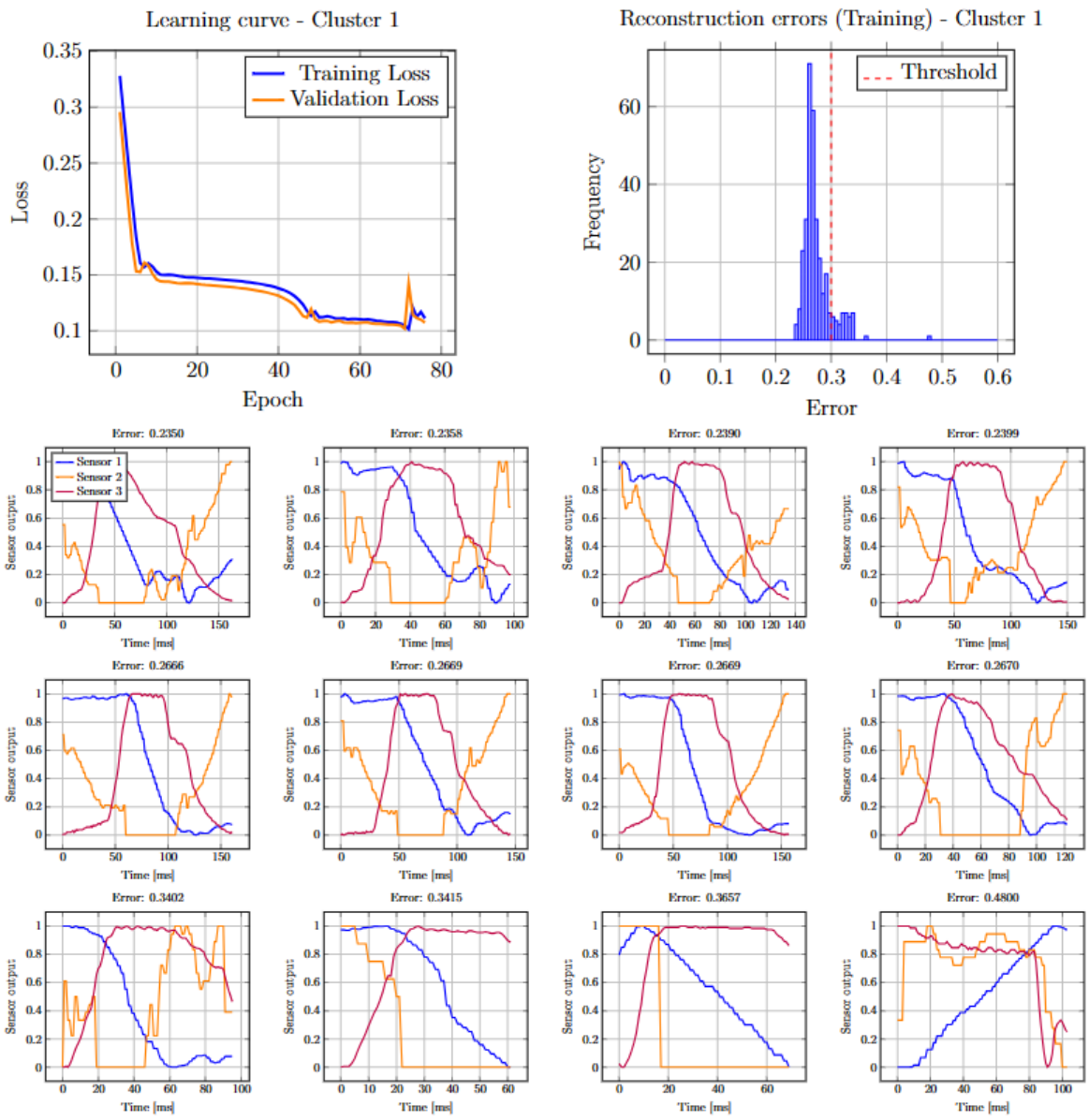


Figure A.19: Training results from the LSTM autoencoder when applied to gear shift logs within cluster 1.

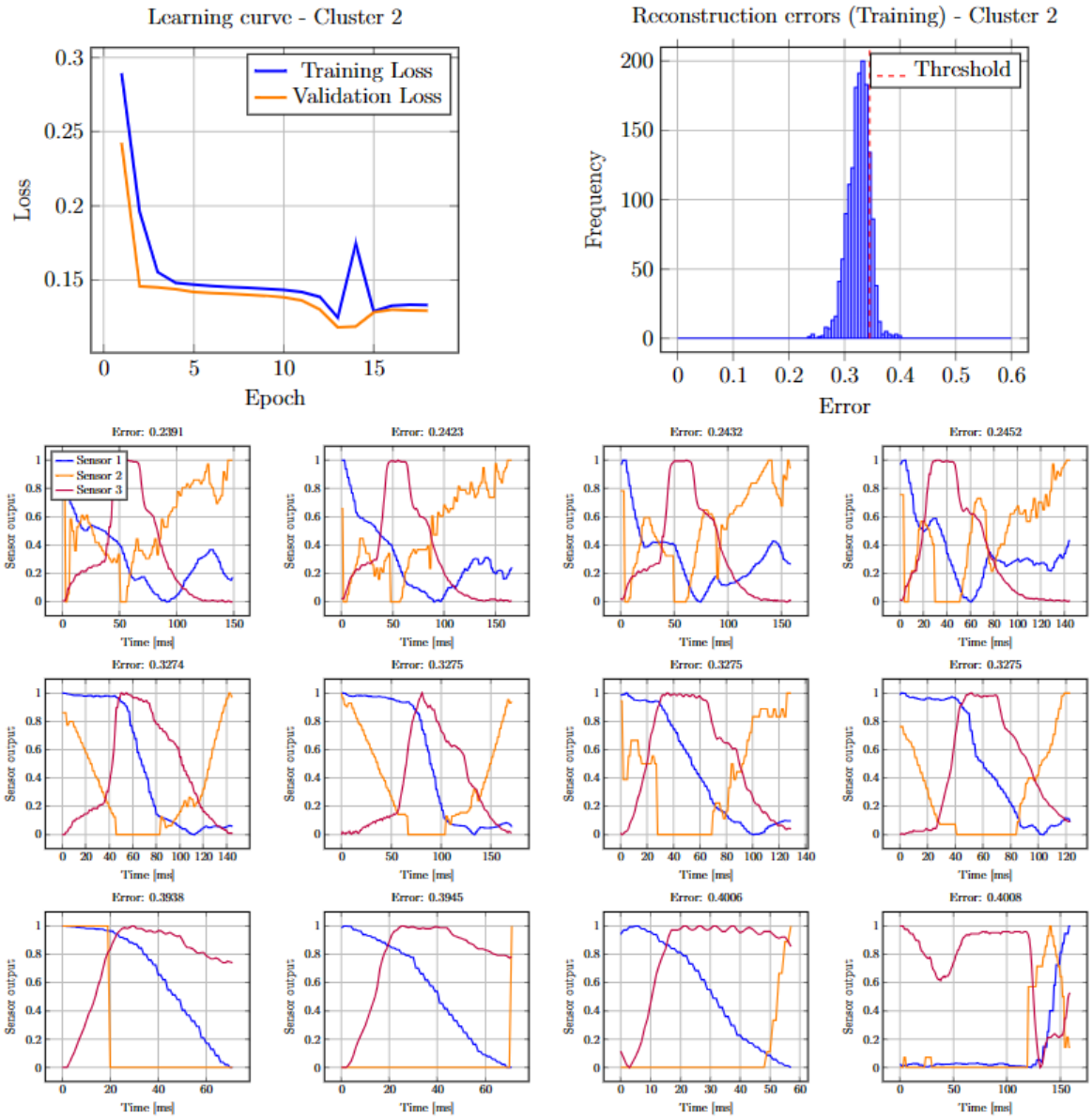


Figure A.20: Training results from the LSTM autoencoder when applied to gear shift logs within cluster 2.

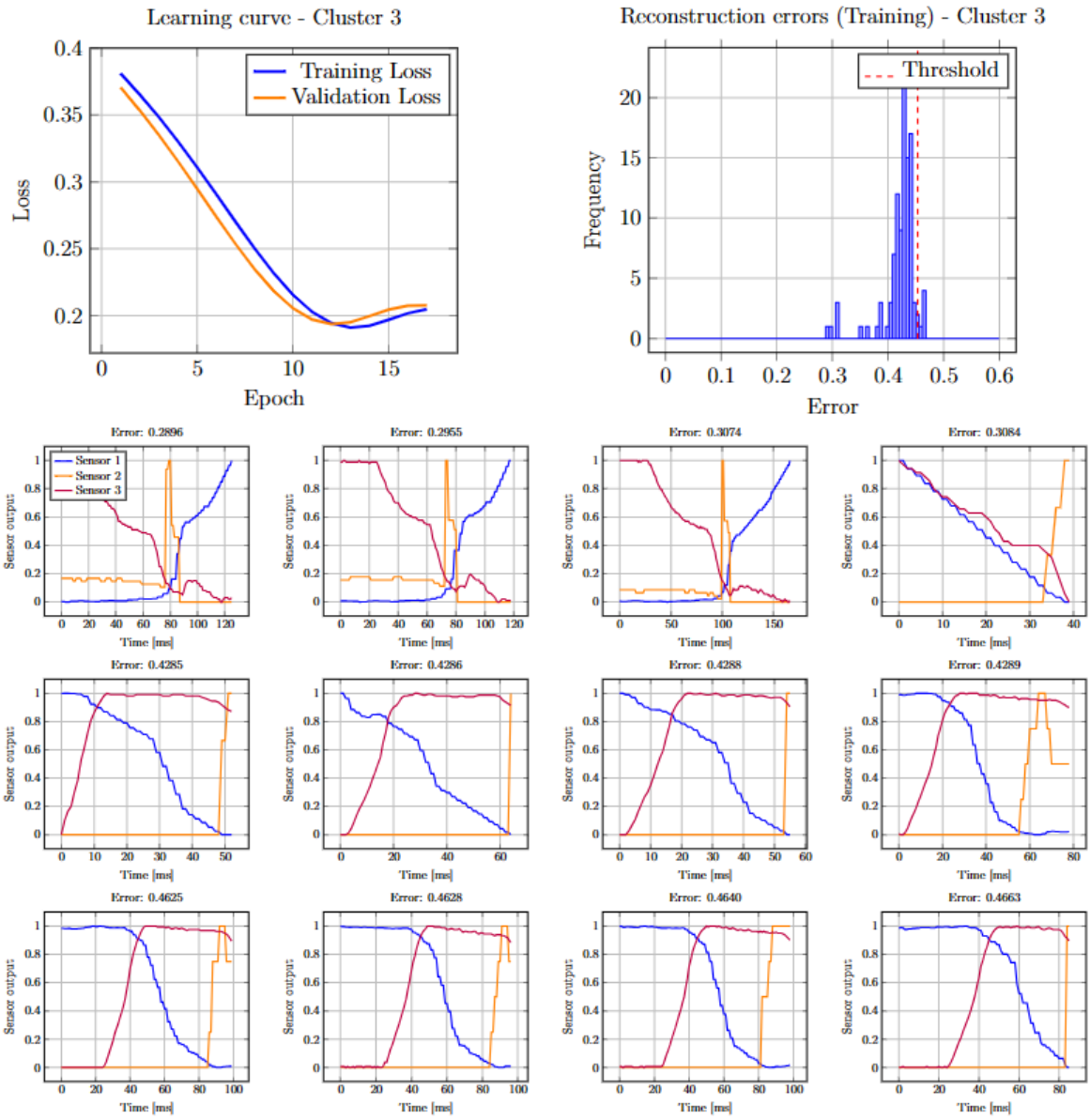


Figure A.21: Training results from the LSTM autoencoder when applied to gear shift logs within cluster 3.

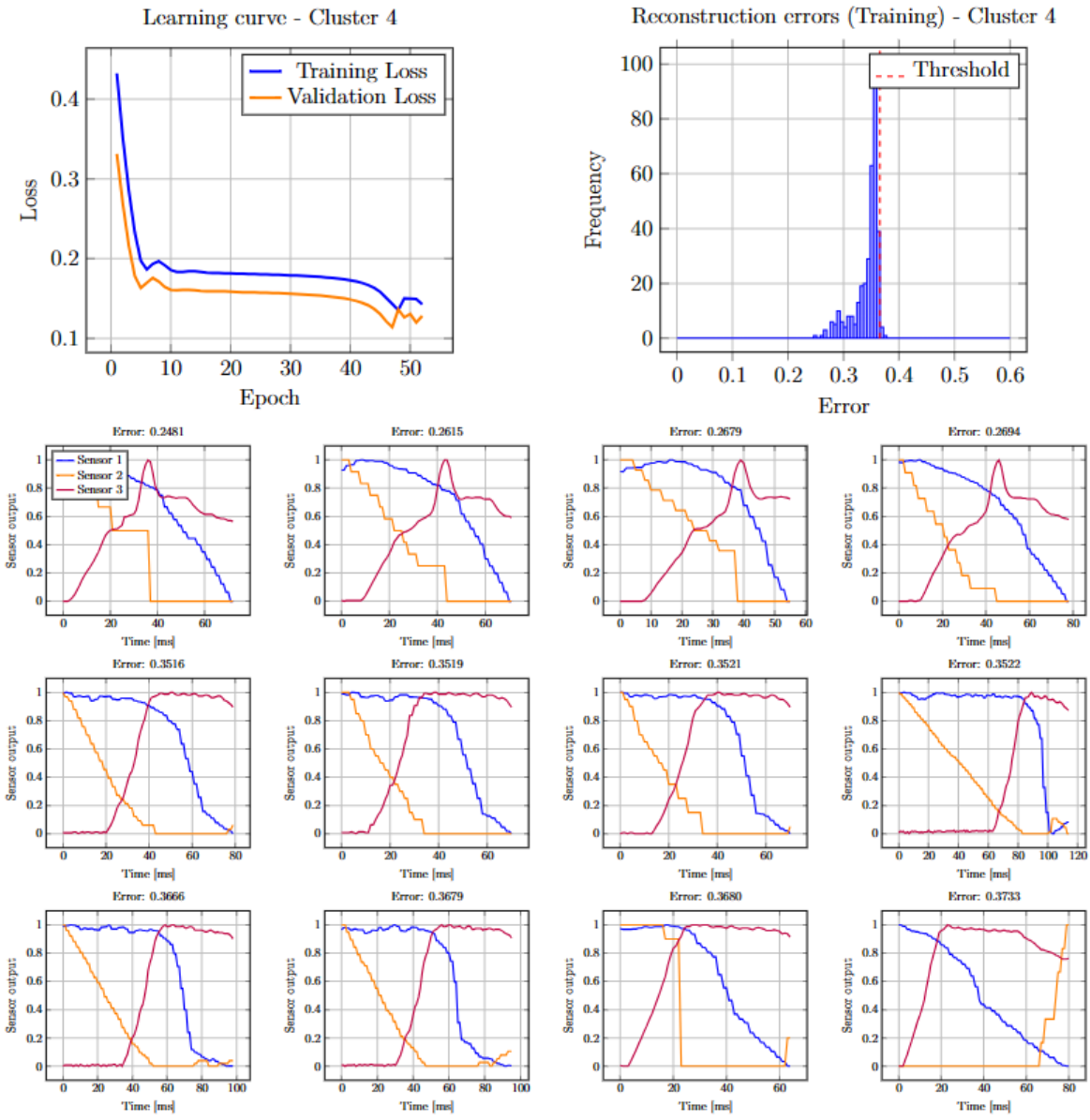


Figure A.22: Training results from the LSTM autoencoder when applied to gear shift logs within cluster 4.

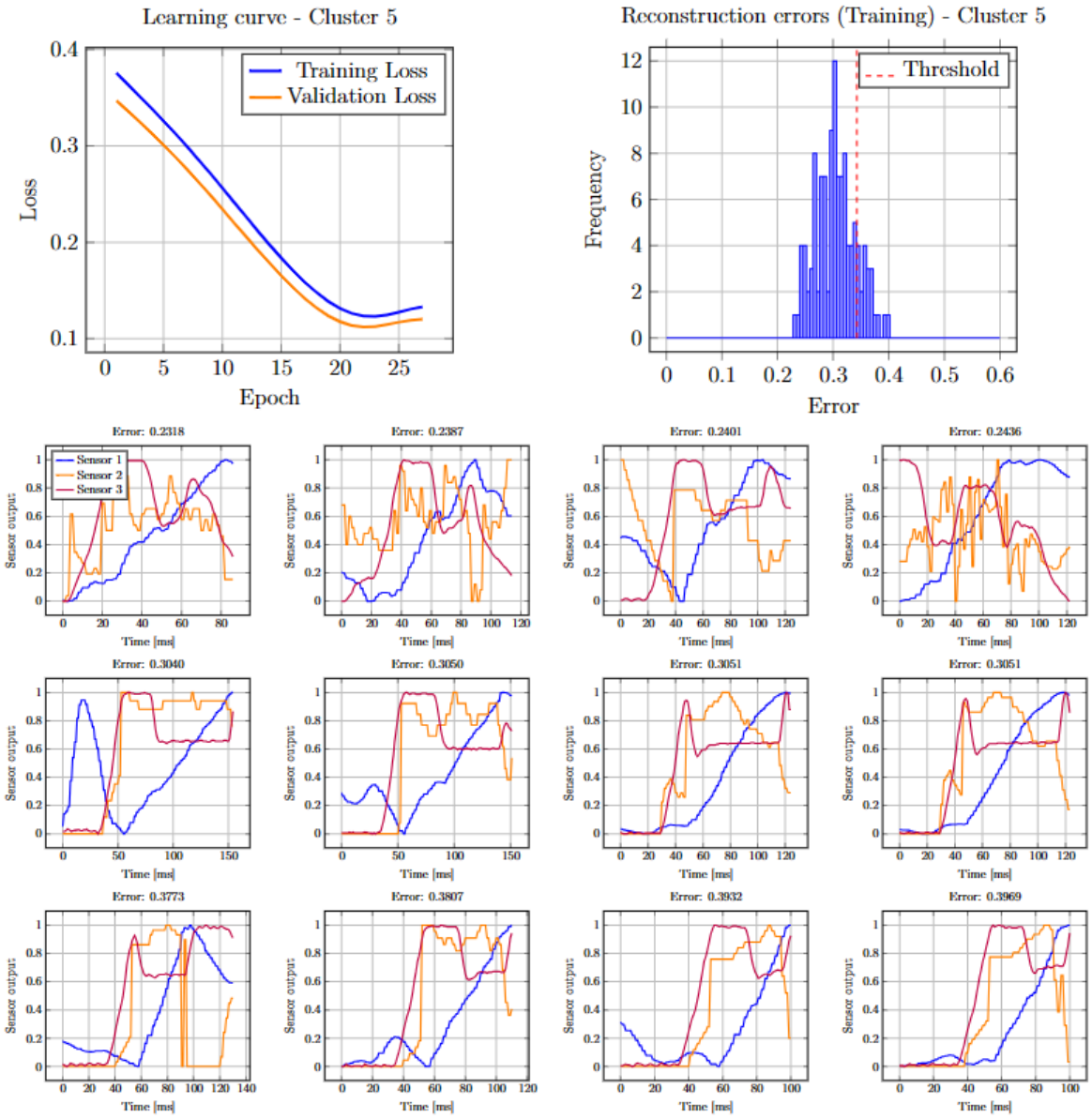


Figure A.23: Training results from the LSTM autoencoder when applied to gear shift logs within cluster 5.

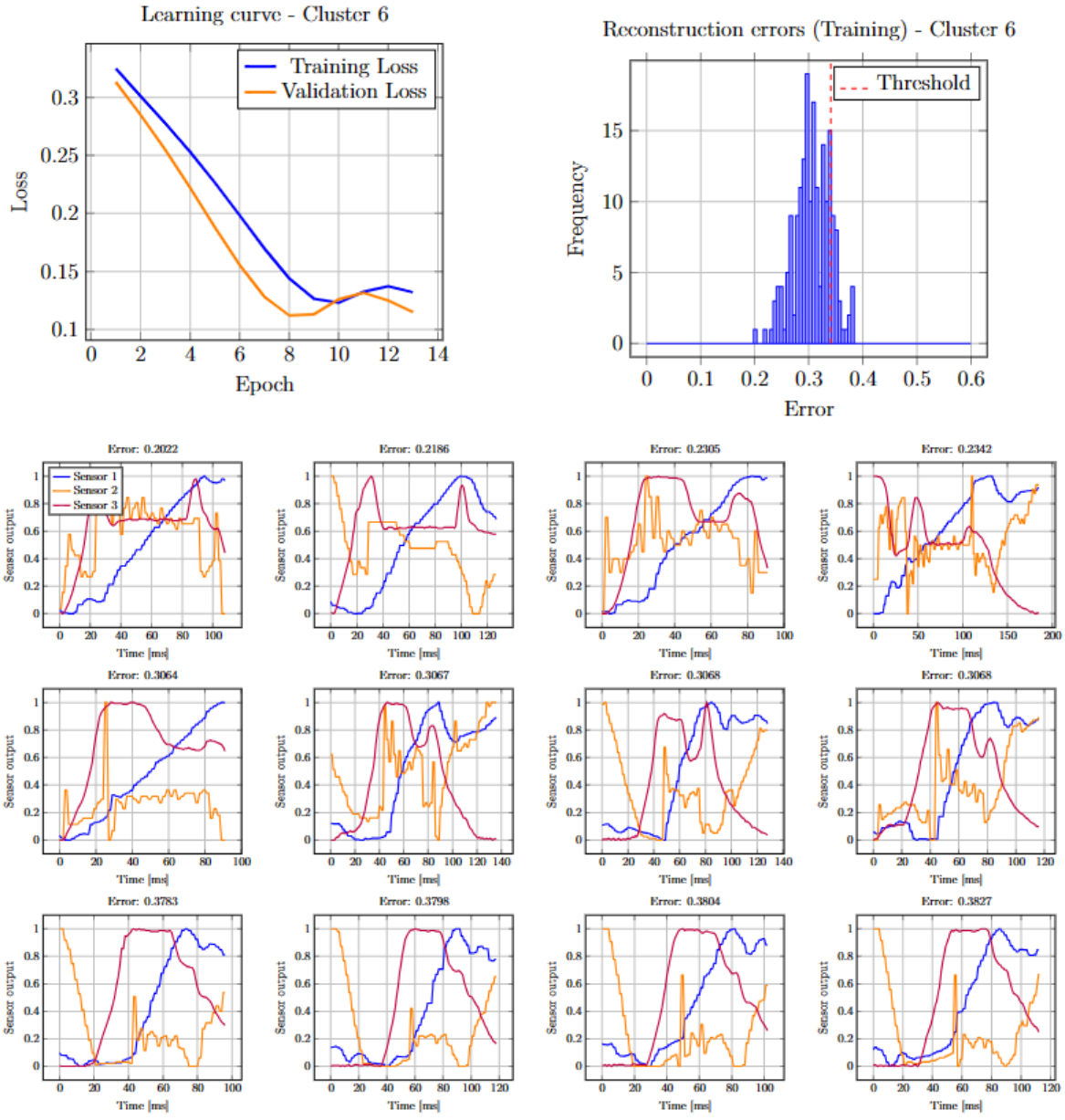


Figure A.24: Training results from the LSTM autoencoder when applied to gear shift logs within cluster 6.

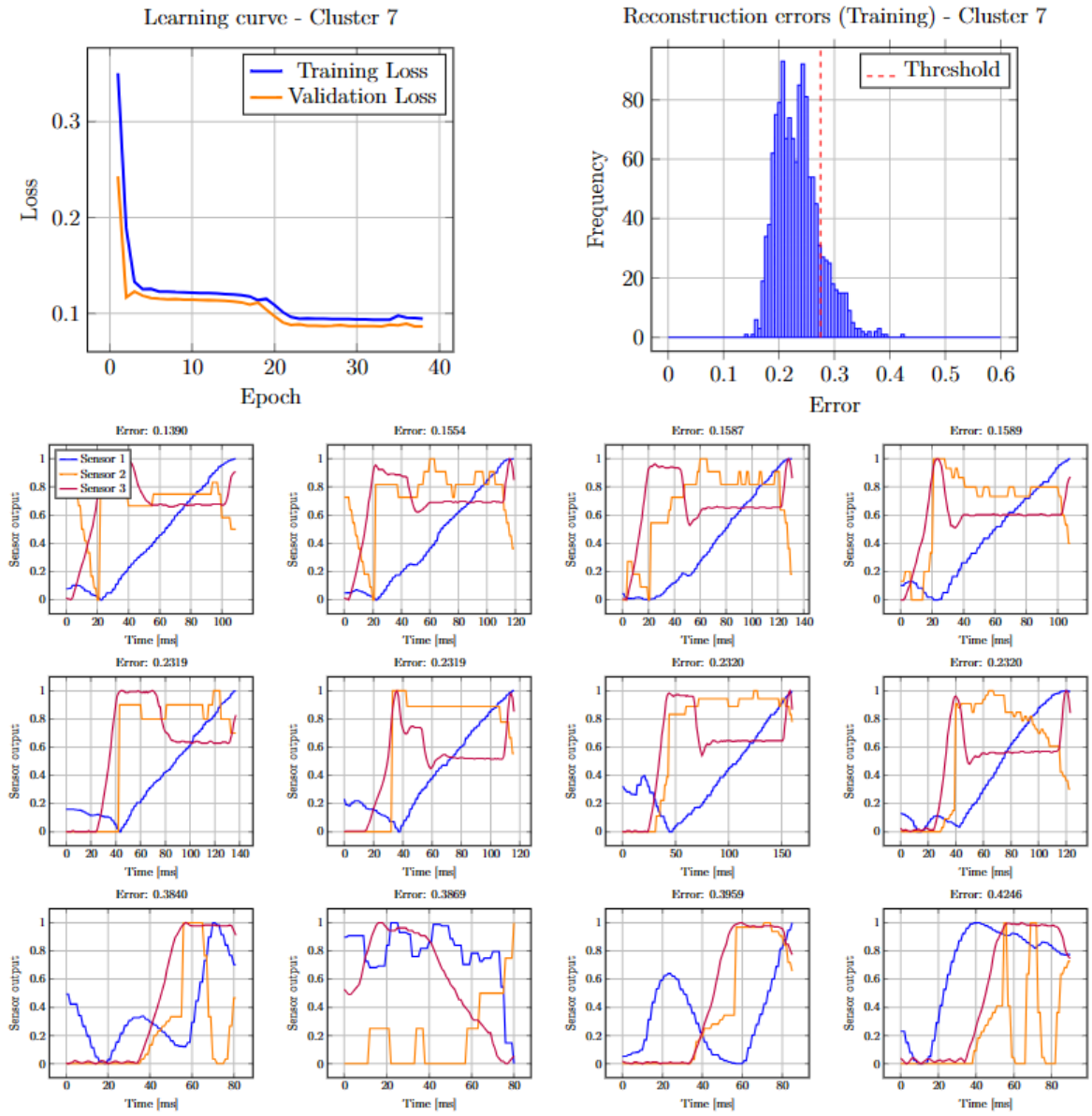


Figure A.25: Training results from the LSTM autoencoder when applied to gear shift logs within cluster 7.

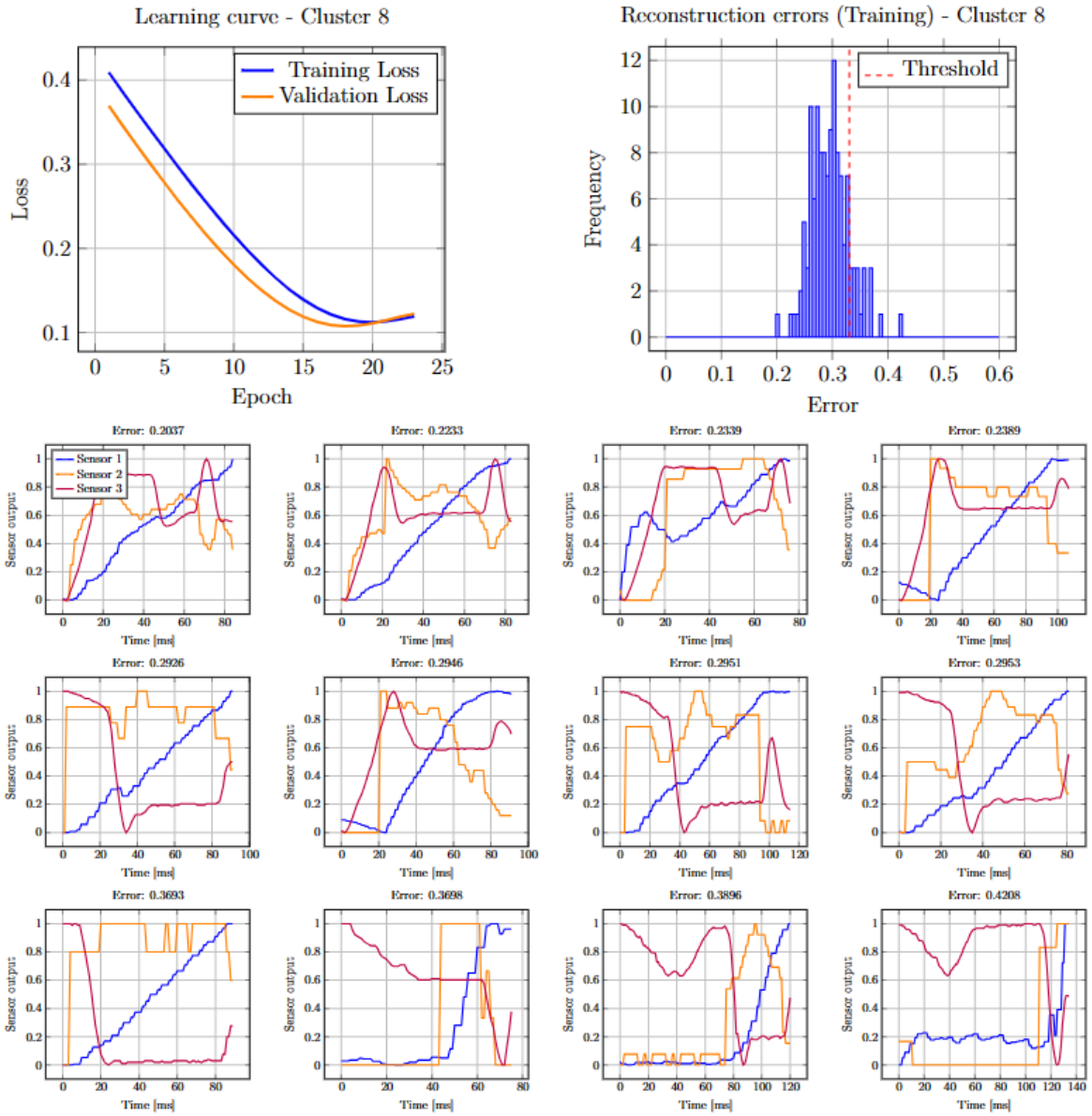


Figure A.26: Training results from the LSTM autoencoder when applied to gear shift logs within cluster 8.

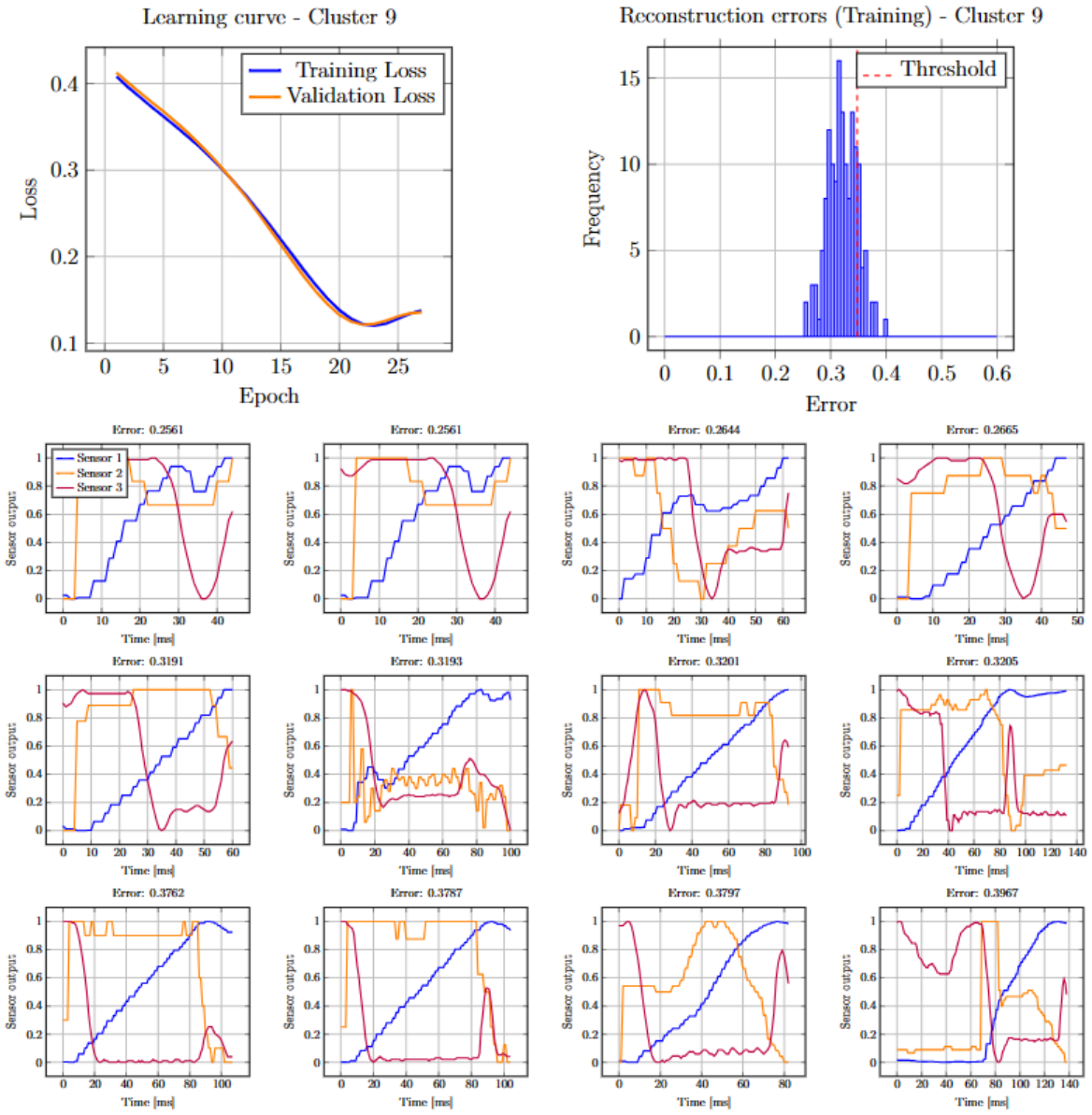


Figure A.27: Training results from the LSTM autoencoder when applied to gear shift logs within cluster 9.

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY