CHALMERS
UNIVERSITY OF TECHNOLOGY

Adaptive Payload Estimation

A Universal Payload Estimation System for Wheel Loaders Using Recurrent Neural Networks and Transfer Learning.

Master's thesis in
Information and communication technology & Complex adaptive systems

MARCUS CARLSSON

ELIN HILJEMARK

DEPARTMENT OF MECHANICS AND MARITIME SCIENCES

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

Master's thesis 2024

Adaptive Payload Estimation

A Universal Payload Estimation System for Wheel Loaders Using Recurrent Neural Networks and Transfer Learning.

MARCUS CARLSSON
ELIN HILJEMARK

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mechanics and Maritime Sciences
Division of Vehicle Engineering and Autonomous Systems
Applied Artificial Intelligence Group
Chalmers University of Technology
Gothenburg, Sweden 2024

Adaptive Payload Estimation
A Universal Payload Estimation System for Wheel Loaders Using Recurrent Neural Networks and Transfer Learning.
MARCUS CARLSSON
ELIN HILJEMARK

Supervisor: Simon Liljeqvist, CPAC Systems AB
Examiner: Peter Forsberg, Department of Mechanics and Maritime Sciences

Master's Thesis 2024
Department of Mechanics and Maritime Sciences
Division of Vehicle Engineering and Autonomous Systems
Applied Artificial Intelligence Group
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Wheel loaders of models Volvo L120Gz and Volvo L150H.

Typeset in LaTeX
Gothenburg, Sweden 2024

Adaptive Payload Estimation
A Universal Payload Estimation System for Wheel Loaders Using Recurrent Neural Networks and Transfer Learning.
MARCUS CARLSSON, ELIN HILJEMARK
Department of Mechanics and Maritime Sciences
Chalmers University of Technology

Abstract

A wheel loader is a versatile construction machine designed for various tasks related to the loading and transport of materials. In many wheel loader applications, accurately knowing the mass of the load being lifted is crucial. For example, when loading materials onto trucks, it is essential to ensure that the load of the truck does not exceed legal weight limits. Therefore, various solutions have been developed to help operators determine the payload mass. However, many existing systems use mechanical equations to estimate the payload, consequently these models are heavily dependent on the geometry of the machine it is implemented for. Using technologies emerging from the field of deep learning, this thesis aimed to investigate the possibilities of a universal payload estimation model. By leveraging real operational data from wheel loaders and formatting it into time sequences, a neural network based on long short-term memory units was constructed. In addition, a classification network was developed based on both convolutional neural network and autoencoder architectures to assess the reliability of data sequences. To generalize the model across different wheel loader models, transfer learning was employed. The results showed that transfer learning enabled the model to accurately estimate payload mass for different wheel loader models. However, the classification network for sensor data reliability was found to be redundant, as it reduced the performance of the payload estimation model. Furthermore, the result also showed a trade-off between the amount of data used for fine-tuning the general model and the accuracy of the model. To further investigate the adaptability and usage of the adaptive payload estimation model, more operational scenarios needs to be tested as well as further examination of the adaptability to different sizes of wheel loaders.

Keywords: Transfer learning, LSTM, payload estimation, wheel loader, deep learning, load weighing, construction equipment.

Acknowledgements

First of all, we would like to thank our examiner, Peter Forsberg, for his support and for inspiring us to think outside the box and to push the boundaries of our work. We would like to thank our supervisor, Simon Liljeqvist, for sharing his domain knowledge and providing invaluable feedback. A special thanks to Anton Widenius for his valuable ideas and insights, and to the entire Load Assist team for their collaboration and support. We are also grateful to CPAC Systems AB for giving us the opportunity to drink a lot of coffee and write this thesis. Finally, we extend our appreciation to Oskar Giljegård and Malte Landgren for their insights and collaboration.

Marcus Carlsson & Elin Hiljemark, Gothenburg, June 2024

Supervisor: Simon Liljeqvist, CPAC Systems AB
Examiner: Peter Forsberg, Department of Mechanics and Maritime Sciences

List of Acronyms

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

1%RER	1% Relative Error Rate
AE	AutoEncoder
BSS	Boom Suspension System
CAN	Controller Area Network
CEC	Constant Error Carousel
CNN	Convolutional Neural Network
FNR	False Negative Rate
FPR	False Positive Rate
GAN	Generative Adversarial Networks
GRU	Gated Recurrent Unit
IMU	Inertial Measurement Unit
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
SVM	Support Vector Machine
TP	Torque Parallel

Nomenclature

Below is the nomenclature of indices, constants, geometrical parameters, and variables that have been used throughout this thesis.

Indices

b	Index representing the boom.
c	Index representing the lift cylinder.
l	Index representing the payload.

Constants

g	The gravitational acceleration.

Geometrical Parameters

ϕ	The angle between the vertical plane, defined with the mounting point of the boom on the body of the wheel loader as origo, and the mounting point of the hydraulic cylinders on the body of the wheel loader.
L_{cm}	The distance from the boom mounting point on the wheel loader body to the hydraulic cylinder mounting pont on the wheel loader body.
L_{cb}	The distance from the mounting point of the boom on the body of the wheel loader to the mounting point of the hydraulic cylinder on the boom.
L_b	The length of the boom.
$\mathbb{P}_{cm}$	Mounting point for the hydraulic cylinder on the body of the wheel loader.
$\mathbb{P}_{cb}$	Mounting point for the hydraulic cylinder on the boom.

Variables

α	The angle between the boom and the direction of the force applied by the hydraulic cylinders.
β	The angle between the direction of the boom and the horizontal plane, defined with the mounting point of the boom as origo.
$\dot{\boldsymbol{\omega}}$	The angular acceleration for the boom around the pivot point.
$\eth_i$	The tangential acceleration, caused by the mass of the object represented by the index i.
$\mathbb{F}_i$	Force generated by the object represented by the index i.
$\mathbb{P}_i$	The center of gravity for the object represented by the index i.
m_i	Mass of the object represented by the index i.
T_i	Torque around the pivot point, generated by the object represented by the index i.
$\mathbb{V}_c$	A vector defining the orientation of the lift cylinder.

Contents

List of Figures

List of Tables

1
Introduction

A wheel loader is a versatile construction machine designed for various tasks related to the loading and transport of materials. Wheel loaders are characterized by their front-mounted attachments, making them suitable for many different tasks, such as earth moving, excavation, pallet handling, log handling, snow removal, and more.

In many wheel loader applications, accurately knowing the mass of the load being lifted is crucial. For example, when loading materials onto trucks, it is essential to ensure that the load of the truck does not exceed legal weight limits. Various solutions and systems have been developed to assist operators in determining the payload mass, including one from CPAC Systems AB. However, the current payload estimation algorithm deployed in this system is highly dependent on precise geometric data of the machine, which restricts the applicability of the system to machines with precisely known geometrical parameters. Furthermore, the system requires a specific lifting motion to generate payload estimates. This requirement can interfere with operational flexibility and may not always be feasible in certain scenarios.

The emerging field of deep machine learning technologies offers a promising prospect for improving payload estimation systems. These technologies could improve the robustness of payload estimates due to their ability to understand the complex dynamics of wheel loaders, allowing for more reliable estimations across a wider range of operational scenarios. Another benefit is their potential to adapt to a variety of machine geometries, possibly even to machines with completely unknown geometrical parameters. This would significantly simplify the customization process for different wheel loader models, and thereby expand the system's versatility.

This thesis will investigate the potential of using deep learning technologies to overcome the current limitations of payload estimation systems, with a particular emphasis on the adaptability to unknown machine geometries. The goal is to develop a flexible, efficient, and universally applicable solution.

1.1 Related work

Payload estimation for construction machinery is an active area of research, with multiple studies focusing on the challenge of accurately predicting payloads for wheel loaders and various other machines. Some of these studies use purely analytical models, among others [1], [2], while other studies, for example [3], [4], employ machine

learning algorithms and neural networks.

The first use of neural networks for payload estimation for wheel loaders was made by Savia. et al. [3]. Their approach involved feeding the neural network with inputs such as hydraulic pressures, boom position, machine inclination angle, and hydraulic oil temperature. Although the authors presented a low predicted error of just over 2%, they noted that a large amount of training data would be required for each individual machine, which makes their system unfeasible in practice.

A common approach for payload estimation is to combine analytical models with machine learning models by feeding the output of the analytical model, along with data about the dynamics of the system, into the machine learning model. This approach was introduced in [5] where an analytical model was utilized to make an initial estimate of the payload mass. The model incorporated several known geometric parameters of the wheel loader linkage system, including the masses and centers of gravity of the linkage components, as well as the force exerted by the tilt cylinder to yield an estimate of the payload mass. To enhance the accuracy of this estimate, a feed-forward neural network was utilized, into which the analytical estimate was an input. Furthermore, the neural network also included the stroke lengths of both the lift and tilt cylinder, the boom's velocity, the differential pressure in the lift cylinder, and data about the accelerations of the machine, captured by an accelerometer. The integration of the neural network resulted in a significant improvement over the purely analytical model. The refined method achieved a worst-case payload estimation error of $\pm$ 1.96% for a full-scale load.

A similar approach was developed for excavators in [6]. This method also began with an approximate analytical model that was later optimized using a machine learning algorithm that worked in unison with the analytical model. In this instance, the algorithm chosen was a Support Vector Machine (SVM). Alongside the output of the analytical model, the SVM was supplied with the stroke of the tilt cylinder, the torque at the bucket joint, and the median payload value estimated during the first second after the bucket was lifted off the ground. The results of this study show that the precision is significantly improved by feeding the analytical estimate into a machine learning algorithm rather than only using analytical estimates, which supports the findings in [5]. A key insight from this study is that it is sufficient to simplify the analytical model by omitting friction effects, and instead, relying on the machine learning algorithm to effectively account for the non-linearities associated with friction.

Furthermore, [4] introduces a multistage progressive model that combines an analytical model, optimized least-squares, and back-propagation neural networks. This method, tested with different loader models, shows promising results in accuracy and robustness, with a notable maximum error of 2%.

All of the previously mentioned articles utilize feed-forward neural networks and SVMs. Although these models proved to be effective, they did not consider temporal

dependencies within the data. In [7], the subject of payload estimation for forestry cranes was explored using a Long Short-Term Memory (LSTM) network. This type of network is capable of learning long-term dependencies, making it suitable for time-series data, such as sensor signals from loading cycles. The methodology included the automatic detection of a load cycle and the collection of sensor data, encompassing hydraulic pressure, oil temperature, crane angles, grapple speed, and acceleration, as well as operator inputs recorded from the machine's joysticks. The study findings indicate an average full-scale error of 1.5% for a single load cycle, demonstrating the efficacy of LSTM networks, and more broadly the potential for machine learning models that can understand temporal relationships in time series data.

1.2 Purpose and research questions

The purpose of this thesis is to develop a payload estimation system for wheel loaders that can be fitted to any wheel loader, along with a certain suite of sensors. The performance metric for the system is the payload estimation accuracy for a full-scale load. The goal is to develop a system that can estimate the payload with an average error of less than 1% of a full-scale load, where a full-scale load is the maximum lifting capacity of the machine.

The thesis aims to provide new knowledge in the field of deep learning applications for industrial machinery. It will investigate the feasibility of using neural networks to estimate the payload mass for wheel loaders. Although neural networks have been researched and deployed for this purpose before, their ability to handle unknown machine geometries and to provide a universal solution for payload estimation remains less studied.

The findings may have broader applications than only for wheel loaders, potentially extending to other machinery such as excavators, articulated haulers, truck cranes, forestry equipment or other kinds of load-carrying hydraulic machinery.

When developing the system, the following research questions will be addressed:

- How can a universal payload estimation system for wheel loaders be implemented using deep learning technologies?
- To what extent can payload estimation be achieved with limited or no knowledge of a wheel loader's specific geometrical parameters?
- How can a deep learning model be implemented to minimize the training data required for each unique machine?

1.3 Scope

The task is to research and develop a deep learning strategy able to estimate the mass of the payload within a 1% error margin relative to a full-scale load. In order to properly study the research questions the following limitations have been made;

- The thesis will only consider wheel loaders and no other types of heavy equipment.
- The system is limited to using only the sensors used in the current load estimation system by CPAC Systems AB.
- The data used will only be collected from wheel loaders from the manufacturer Volvo.
- All evaluations will be conducted offline on recorded logs.
- The computational complexity and the feasibility of running the final model in a wheel loader are not considered.

2
Theory

This chapter will cover the essential knowledge of the different fields needed to understand the algorithms developed.

2.1 Recurrent Neural Networks

There exist multiple different architectures for neural networks, each with its own strengths and weaknesses. One that has been shown to be useful for managing time series data is the Recurrent Neural Network (RNN) [8]. The distinctive feature of RNNs lies in their recurrent connections which enables them to maintain a memory of previous inputs by incorporating the output of a hidden layer from one time step as part of the input for the same layer in the next time step. This allows RNNs to understand temporal relationships within the data, which makes them particularly well suited for tasks that involve the processing of sequential data, such as time series data [9].

A basic RNN structure might consist of three layers: an input layer, a recurrent hidden layer, and an output layer. Each layer has connections to the next layers, and the hidden layer also has units that connect to each other over time, through recurrent connections. This enables the network to leverage previous information given to the network approximately ten time steps back in time [10].

2.1.1 Difficulties with RNNs

The training procedure for an RNN comes with difficulties, as the probabilities of vanishing or exploding gradients are significant [11]. Consider an RNN unfolded over time and the process of weight updates through the backpropagation through time algorithm. Let the time steps be discrete values indexed by the variable τ, and define an epoch as the duration from an initial time t' to a concluding time t. Define U as the set comprising all non-input units within the network, and $\mathbf{f}_u$ as the nonlinear, differentiable squashing function for a unit $u \in U$. Furthermore, let $z_u(\tau)$ represent the weighted input to unit u, such that $z_u(\tau+1) = \sum_v wu, vy_v(\tau) + \sum_i w_{u,i}y_i(\tau+1)$, where v denotes the precursors of unit u, i indicates the input units, and $w_{i,j}$ denote the weight between units i and j. Therefore, the output of the unit u at time τ can be expressed as follows [9]:

$$y_u(\tau) = \mathbf{f}_u(z_u(\tau)). \tag{2.1}$$

Consequently, the mean square error of output unit k at time τ can then be expressed

as [11]:

$$\vartheta_k(\tau) = \mathbf{f}_k^{'}(z_k(\tau))(d_k(\tau) - y_k(\tau)), \tag{2.2}$$

with $d_k(\tau)$ as the target value for unit k at time τ. The back propagated error at a non-output unit j is given by:

$$\vartheta_j(\tau) = \mathbf{f}_j^{'}(z_j(\tau)) \sum_i w_{i,j}\vartheta_i(\tau + 1). \tag{2.3}$$

This results in the total weight update being $\alpha\vartheta_j(\tau)y_l(\tau - 1)$, with α denoting the learning rate and l representing a unit connected to unit j [11].

Let the RNN be fully connected and the non-input units be indexed from 1 to n. The local error between two arbitrary units, u and v, can then be expressed by [11]:

$$\frac{\vartheta_v(\tau - 1)}{\vartheta_u(\tau)} = \begin{cases} \mathbf{f}_v(z_v(\tau - 1))w_{u,v}, & q = 1 \\ \mathbf{f}_v(z_v(\tau - q)) \sum_{l=1}^{n} \left(\frac{\partial\vartheta_l(\tau - q + 1)}{\partial\vartheta_u(\tau)} w_{l,v} \right), & q > 1, \end{cases} \tag{2.4}$$

where q denotes the number of steps back in time. By letting $l_0 = u$ and $l_q = v$, the local error becomes [11]:

$$\frac{\vartheta_v(\tau - 1)}{\vartheta_u(\tau)} = \sum_{l_1=1}^{n} \cdots \sum_{l_{q-1}=1}^{n} \prod_{m=1}^{q} \mathbf{f}_{l_m}^{'}(z_{l_m}(\tau - m))w_{l_m,l_{m-1}}. \tag{2.5}$$

If $|\mathbf{f}_{l_m}^{'}(z_{l_m}(\tau - m))w_{l_m,l_{m-1}}| > 1$ the product will eventually blow up, and if $|\mathbf{f}_{l_m}^{'}(z_{l_m}(\tau - m))w_{l_m,l_{m-1}}| < 1$ it will eventually disappear. Leading to either exploding or vanishing gradients, which will induce unwanted behaviour [11]. To address these issues, LSTM-RNNs and Gated Recurrent Unit RNNs (GRU-RNNs) were developed [10]. Since GRUs are similar to LSTMs, the difference being less external gating signals [12], the theory will cover only the relevant mathematical background for the LSTM-RNNs.

2.1.2 Long Short-Term Memory Recurrent Neural Networks

The architecture of LSTM networks introduces modifications to the traditional RNN architecture, primarily through the introduction of the Constant Error Carousel (CEC) and memory blocks [9]. The CEC is constructed to counter the problem of vanishing gradient, by ensuring that the local error feedback within a unit over time remains constant. This is achieved by setting the derivative of the activation function of the unit, $\mathbf{f}_j^{'}$, in relation to its weight, $w_{j,j}$, such that

$$\mathbf{f}_j^{'}(z_j(\tau))w_{j,j} = 1 \implies \mathbf{f}_j(\tau) = k\tau + m. \tag{2.6}$$

As a result, the function $\mathbf{f}_j(\tau)$ has to adopt a linear form, $k\tau + m$, to maintain this condition, thus keeping the activation function of the unit constant over time, as seen in Equation 2.7 [11].

$$y_j(\tau + 1) = \mathbf{f}_j(z_j(\tau + 1)) = \mathbf{f}_j(w_{j,j}y_j(\tau)) = y_j(\tau). \tag{2.7}$$

Although the CEC ensures that the error feedback within a unit is constant, it also interferes with other units in the network. This ensures that unwanted behavior does not emerge from these external interactions, as the CEC must be able to take into account these additional weighted inputs and outputs [9]. This challenge is addressed by introducing memory blocks, each consisting of a CEC, an input gate, an output gate, and a forget gate. A schematic diagram of a memory block is shown in Figure 2.1.

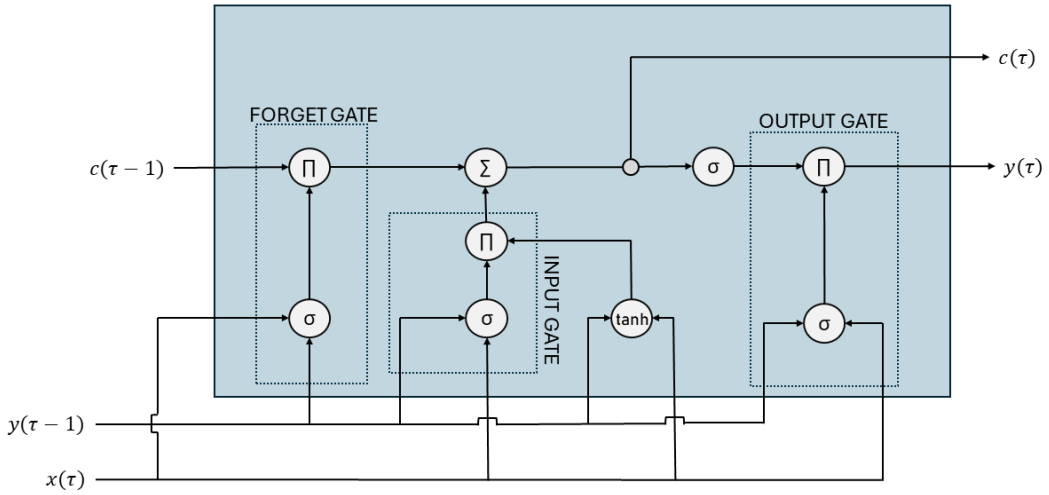


Figure 2.1: A schematic diagram of a LSTM memory block. In this figure, $c(\tau)$ represents the state of the memory block at time τ, $y(\tau)$ denotes the output of the memory block at time τ and $x(\tau)$ indicates the new input to the memory block at time τ

The input gate consists of threshold units with sigmoid activation functions, which regulate the signals transmitted from the rest of the network to the memory block. They modulate these signals by scaling the inputs according to the activation function's value, thereby shielding the CEC from external disturbances [9]. The output gates work in the same manner, but focus on protecting the network from any disruptive influences originating within the memory block itself [11]. The forget gate dictates the retention or discarding of information from the memory of the cell. This helps the network drop old, unneeded data and thereby improve performance. The output of the forget gate ranges from 0 to 1 for each element in the cell state, with 0 instructing the cell to discard the information and 1 to retain it, allowing the LSTM network to dynamically manage its memory and improve its performance for sequential data [9].

2.2 Anomaly detection

The definition of an anomaly in [13] is: "*An anomaly is an observation which deviates so much from other observations as to raise suspicions that it was generated by a different mechanism*". Anomaly detection is the process of identifying such deviations by comparing them with the normal behavior of a system [14]. There exist multiple different approaches for anomaly detection, which can generally be categorized into three main types; reconstruction-based methods, prediction-based methods, and classification-based methods [15], [16].

In reconstruction-based methods, models are trained to replicate the input data under normal operating conditions. The presence of an anomaly causes a significant increase in the reconstruction error. An effective model in this category is the AutoEncoder (AE), which consists of two parts, an encoder and a decoder. The encoder compresses the data into a lower-dimensional representation, and the decoder attempts to decode the data from the low-dimensional representation back to the original format of the input. An example of an AE applied for anomaly detection was presented in [17], where an AE consisting of stacked convolutional layers and LSTM layers was used to successfully detect anomalies in video surveillance monitor data.

In prediction-based methods, the models are also trained to learn the behavior of the systems normal operations. However, here the models are trained to predict the next value and then calculate the prediction errors against the actual outcomes [15]. Various models are applicable here, including LSTM networks, hierarchical temporal memory networks, and SVM [18].

Lastly, classification-based methods classify the data as either normal or anomalous. The models are trained to find a decision boundary between the samples of normal behavior and erroneous behavior. This can be done by training the models only on data from normal behaviors and letting the model assume that everything outside of the learned distribution is anomaly [16]. Some of the most commonly used methods for this type of anomaly detection include one-class SVM, random forest, and k-nearest neighbors [16].

2.3 Data augmentation

Limited access to training data is a common issue when developing neural networks, which often results in suboptimal model performance [4]. A widely adopted solution to this problem is data augmentation, in which synthetic data is generated to enhance the size of the dataset. There are multiple different techniques for synthetic data generation; these techniques can generally be classified into two main categories, traditional algorithms and machine learning model-based approaches [19].

Traditional algorithms cover the methods where synthetic data is generated by us-

ing transformations on already existing data, for example, cropping, scaling, and jittering [19]. Jittering, for example, involves adding random noise to the original data points to create a new noisier dataset. By assuming that there exists noise in the original data, jittering generates new samples by simulating this noise. For an original signal represented as $x = \{x_1, ..., x_n\}$, jittering results in a new signal $x' = \{x_1 + \epsilon_1, ..., x_n + \epsilon_n\}$, where ϵ_i represents the noise added to each data point. However, this method can introduce unwanted variability. In [20] jittering did not provide the model with improved performance due to major fluctuations in the data introduced by the added noise of the jittering.

The other category uses machine learning models to generate entirely new data samples. The most commonly used model is Generative Adversarial Networks (GAN). GANs operate in a game-theoretic way in which two neural networks, the generator and the discriminator, compete in a min-max zero-sum game [21]. The generator creates new data points from random noise, typically following a Gaussian distribution [22]. The generator is trained to generate a distribution of data that mimics the distribution of actual data. The Discriminator is trained to predict if data belong to the real distribution or if it is synthetically made. The classifications performed by the discriminator is providing feedback to the generator to refine its output. This training process enhances the generator's ability to produce data points that are indistinguishable from real data [22].

2.4 Transfer learning

The duration and complexity of the learning process for a new subject can vary significantly due to differing prerequisites. For humans, this process can be accelerated by leveraging prior knowledge from related areas. For example, knowledge about how to play the transverse flute can shorten the process of learning to play the saxophone, as many of the fingerings are similar between the two instruments. This ability can also be used to enhance the performance of deep learning models, where techniques such as transfer learning can apply previously acquired knowledge to other, different but related tasks [23].

In transfer learning, a domain $\mathcal{D}$ is defined, which consists of two different components, a feature space $\mathcal{X}$ and a marginal probability function $P(X)$, where $X = \{x_1, ..., x_n\}$ in which $x_j \in \mathcal{X} \quad \forall j \in [1, n]$ [24]. A task associated with a domain consists of a label space $\mathcal{Y}$ and an unknown predictive function $\mathbf{f}$. The predictive function $\mathbf{f}$ is not known, but will be learned through training data, which consist of pairs $\{x_i, y_i\}$ where $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$ [23]. Transfer learning then aims to improve the learning of a predictive function $\mathbf{f}_T$ in a target domain $\mathcal{D}_T$ using the knowledge acquired from a source domain $\mathcal{D}_S$ [23].

Pan and Yang [24] categorize transfer learning into three types based on the relationship between the source and target label spaces:

- Inductive Transfer Learning: Both the source and target tasks are similar,

with labeled data available for the target. Labeled target data is used to induce a transfer function $\mathbf{f}_T$ to use in the target domain. If the source domain consists of a lot of labeled data, the knowledge from the source domain will be used to improve the accuracy of $\mathbf{f}_T$ in the target domain.

- Transductive Transfer Learning: The source and target task is the same while the source and target domains differ. Here, no labeled data in the target domain is available.

- Unsupervised transfer learning: Used when the source and target tasks are different but related. Unlike inductive transfer learning, this category concerns the case in which neither of the source or target domain has access to labeled data [23].

Inductive transfer learning can be implemented using pre-trained networks and fine-tuning specific components to obtain $\mathbf{f}_T$. Gupta et al. explored the advantages of employing pre-trained RNNs for binary classification tasks in their study [25]. The result indicated that models using this transfer learning approach outperformed those trained only on the specific task. Similar results have been observed in other research areas.

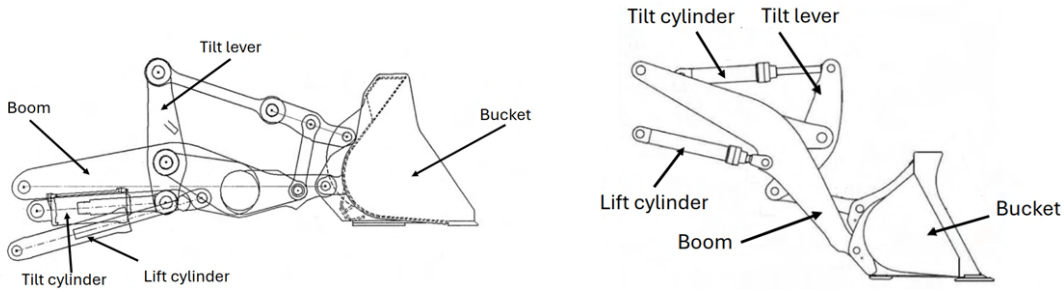
However, to adapt the network to a new task, it needs to be adjusted to the specific domain. This raises the question of how to effectively retrain the model [26]. Different layers within a model tend to capture various aspects of the data; for example, in a CNN the initial layers generally retain more generic features, while the top layers are more likely to learn more specific, task-related features. [27] investigated the transferability of features across different layers in a neural network. The conclusions were that there are two distinct issues that negatively affect the transferability: "*optimization difficulties related to splitting networks in the middle of fragilely co-adapted layers and the specialization of higher layer features to the original task at the expense of performance on the target task*". In other words, in order to ensure effective transfer learning, the adaptation of the network to its new task needs to be done in such a way that layers heavily dependent on each other do not get split up, and in a manner that specializes the top layers to the new target task. A commonly used method that does not create any of the mentioned problems is freezing the lower layers of the pre-trained network and fine-tuning the upper layers [26].

2.5 Wheel loader geometry

To understand the problem of payload estimation for wheel loaders, it is essential to have fundamental knowledge about the design of a wheel loader, especially with regard to its hydraulic system and the kinematics of the linkage.

A wheel loader is generally a machine that is steered with an articulating joint between a front and rear frame. There is a hydraulically actuated kinematic linkage attached to the front frame, on which an attachment is mounted [5]. There are

primarily two types of linkage configurations: the Z-bar linkage and the Torque Parallel (TP) linkage, each with distinct geometrical characteristics, as illustrated in Figure 2.2.



(a) TP linkage **(b)** Z-bar linkage

Figure 2.2: The two common types of linkage geometries for wheel loaders

Central to these configurations is the boom, fixed to the front frame through a pivot joint, which is maneuvered by the lift cylinders to raise or lower. The bucket's tilting action is controlled by the tilt cylinder, yet the specific linkage geometry that facilitates this tilting differs between the Z-bar and TP configurations.

The advantage with a TP linkage is its self-adjusting capabilities, which eliminate the need for constant manual adjustments to maintain a consistent bucket angle [28]. This feature is particularly advantageous in scenarios that require the full range of the lifting capacity, such as truck loading operations. However, the Z-bar linkage, while offering less self-adjustment, provides a higher torque in the digging position, making it more suitable for excavation and material rehandling tasks [5].

On the subject of payload estimation, the TP linkage presents an additional advantage. Specifically, the center of gravity for the bucket and its load remain relatively consistent across different boom angles. This consistency simplifies the kinematic equations, allowing for the assumption that the bucket's center of gravity is in the same place as the boom's endpoint in the x- and y-coordinates, with an offset in the z-coordinate, thereby simplifying the calculations involved in the payload estimation.

Most wheel loaders are equipped with a Boom Suspension System (BSS), designed to mitigate the effects of vibrations when driving on rough surfaces. This system enhances the stability of the machine and improves operator comfort. Damping is typically achieved through the use of passive hydro-pneumatic accumulators connected to boom lift cylinders [29].

3
Methods

The following chapter covers how the payload estimation models were constructed and implemented to receive the results presented in Chapter 4. The development of the final model was divided into three steps. Initially, a purely analytical model was derived; this was done based on insights from related work referenced in Section 1.1, which highlighted the efficiency of using an analytical foundation to simplify the problem space of a neural network. Following the mechanical model, a deep learning model was developed specifically for a single machine type, which utilized the output of the mechanical model as input. The culmination of the development process was the creation of a generalized model, trained on multiple different machines, and subsequently fine-tuned for a specific machine. In addition, a classification network was developed to classify the reliability of the data. Each of these steps, as well as the data acquisition and processing, is described in the following sections.

3.1 Mechanical model

The first step was to create a purely mechanical model based on rotational equilibrium principles. The purpose of this model was on one hand to get a deep understanding of the problem of payload estimation, but also because this model would later serve as an input to the deep learning models.

3.1.1 Calculation of torque equilibrium

The system was modeled as a homogeneous rod, representing the boom and an attached payload. The boom, with mass m_b, acts as a fixed lever arm at one end, with its center of gravity located at the midpoint. The payload, with mass m_l, applies a force at the outer end of the boom, and the hydraulic cylinder lifts the boom with a force F_c. In this simplification, the masses of the attachment, the attachment bracket, the tilt linkage, as well as the lubrication system and hydraulic hoses, mounted onto the boom, were integrated into the mass of the boom. A schematic illustration of the system is shown in Figure 3.1.

To find an equilibrium, the torques generated by the gravitational forces acting on the boom, from the payload, T_l, and the weight of the boom itself, T_b, as well as the torque applied by the hydraulic cylinder, T_c, were considered. The system is assumed to be in rotational equilibrium when the sum of these torques around the boom pivot point is zero. Mathematically, this condition is expressed as follows:

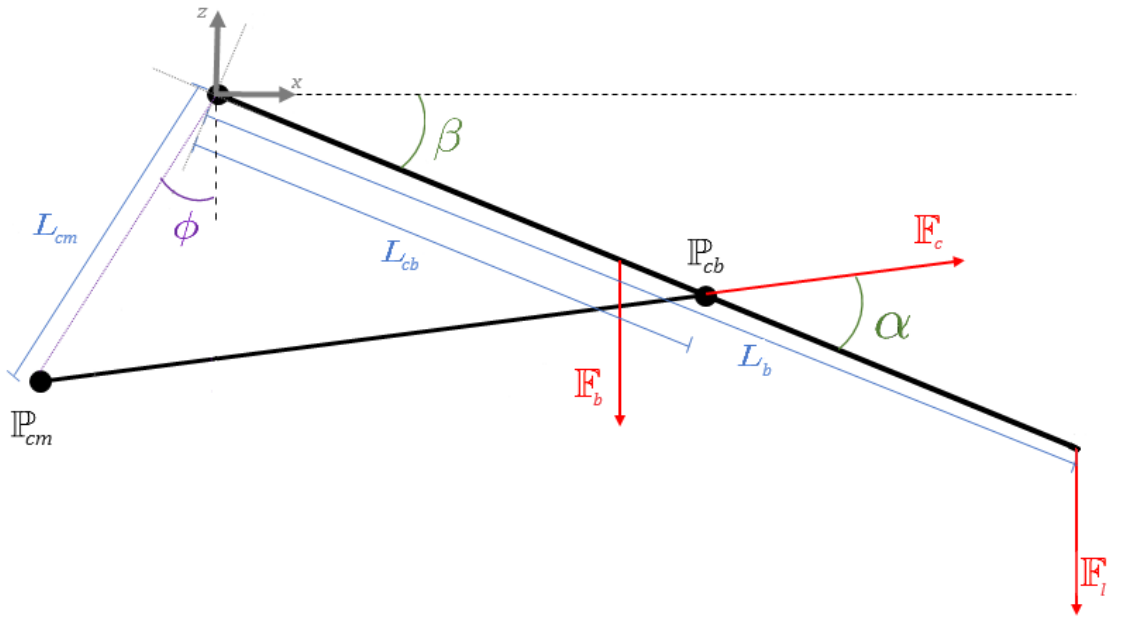


Figure 3.1: The forces acting on the simplified model of the linkage. The boom is represented by a homogeneous rod and the payload is modelled to act only on the outer edge of the rod. Note that the forces are not illustrated in scale to each other.

$$0 = T_b + T_l + T_c. \tag{3.1}$$

For the boom, the gravitational force acts at a distance $\frac{L_b}{2}$ from the pivot point, where L_b is the length of the boom. For the payload, the force is applied at the end of the boom, at a distance L_b from the pivot. Using a coordinate system with the origin at the pivot point, and defining β as the angle between the boom and the horizontal plane, the centers of gravity for the two forces will be:

$$\mathbb{P}_l = [L_b \cos\beta, 0, -L_b \sin\beta], \tag{3.2}$$

$$\mathbb{P}_b = \left[\frac{L_b}{2}\cos\beta, 0, -\frac{L_b}{2}\sin\beta\right]. \tag{3.3}$$

Next, the angular acceleration for the boom around the pivot point, $\dot{\boldsymbol{\omega}}$ is introduced, which is given by:

$$\dot{\boldsymbol{\omega}} = [0, -\dot{\omega}, 0]. \tag{3.4}$$

The tangential accelerations $\eth$, due to the angular acceleration for the payload and the boom's center of gravity are calculated using the cross product:

$$\eth_l = \dot{\boldsymbol{\omega}} \times \mathbb{P}_l = [\dot{\omega} L_b \sin(\beta), 0, \dot{\omega} L_b \cos(\beta)], \tag{3.5}$$

$$\eth_b = \dot{\boldsymbol{\omega}} \times \mathbb{P}_b = \left[\dot{\omega}\frac{L_b}{2}\sin(\beta), 0, \dot{\omega}\frac{L_b}{2}\cos(\beta)\right]. \tag{3.6}$$

The effective forces due to these accelerations are:

$$\mathbb{F}_l = m_l(g + \eth_l) = m_l[0, 0, -g] + m_l[\dot{\omega} L_b \sin(\beta), 0, \dot{\omega} L_b \cos(\beta)], \tag{3.7}$$

$$\mathbb{F}_b = m_b(g + \eth_b) = m_b[0, 0, -g] + m_b \left[\dot{\omega} \frac{L_b}{2} \sin(\beta), 0, \dot{\omega} \frac{L_b}{2} \cos(\beta)\right]. \tag{3.8}$$

Such that:

$$\mathbb{F}_l = [m_l \dot{\omega} L_b \sin(\beta), 0, -m_l g + m_l \dot{\omega} L_b \cos(\beta)], \tag{3.9}$$

$$\mathbb{F}_b = \left[m_b \dot{\omega} \frac{L_b}{2} \sin(\beta), 0, -m_b g + m_b \dot{\omega} \frac{L_b}{2} \cos(\beta)\right]. \tag{3.10}$$

The torques can now be calculated as the cross product of the position vectors and the force vectors:

$$T_l = \mathbb{P}_l \times \mathbb{F}_\mathrm{l} = [0, -L_b m_l g \cos(\beta) + L_b m_l \dot{\omega} L_b \cos^2(\beta) - L_b m_l \dot{\omega} L_b \sin^2(\beta), 0], \tag{3.11}$$

$$T_b = \mathbb{P}_b \times \mathbb{F}_\mathrm{b} = [0, -\frac{L_b}{2} m_b g \cos(\beta) + \frac{L_b}{2} m_b \dot{\omega} \frac{L_b}{2} \cos^2(\beta) - \frac{L_b}{2} m_b \dot{\omega} \frac{L_b}{2} \sin^2(\beta), 0]. \tag{3.12}$$

The torque applied by the hydraulic cylinder, T_c, can be expressed as a function of the force vector of the cylinder F_c. The direction and magnitude of this force will depend on the cylinder's orientation and the hydraulic pressures within the cylinder. The orientation depends on the angle of the boom β, as the point where the cylinder is attached to the boom will rotate with the angle of the boom. This point, $\mathbb{P}_{cb}$, will be located at a distance of L_{cb} from the pivot point and rotate around the y-axis. $\mathbb{P}_{cb}$ can be defined in the coordinate system using the rotation matrix for rotation around the y-axis, R_y.

$$\mathbb{P}_{cb}(\beta) = R_y(\beta)[L_{cb}, 0, 0]^T, \tag{3.13}$$

$$= \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{bmatrix} \begin{bmatrix} L_{cb} \\ 0 \\ 0 \end{bmatrix}. \tag{3.14}$$

The other cylinder attachment point, $\mathbb{P}_{cm}$, is located on the machine, at a fixed distance L_{cm} and a fixed rotation ϕ relative to the pivot point:

$$\mathbb{P}_{cm} = [-L_{cm} \sin(\phi), 0, -L_{cm} \cos(\phi)]. \tag{3.15}$$

By subtracting the mounting point on the boom $\mathbb{P}_{cb}$ from the mounting point on the machine $\mathbb{P}_{cm}$, the vector was obtained for the orientation of the hydraulic cylinder $\mathbb{V}_c = \mathbb{P}_{cm} - \mathbb{P}_{cb}$.
The component of the hydraulic cylinder force that creates torque around the pivot point could be found with a scalar projection, using that the dot product between

two vectors can be expressed in terms of the angle between them. Taking the scalar projection of the vector $\mathbb{V}_c$ and the vector between the pivot point and the mounting point of the cylinder to the boom, the angle between the boom and the force vector from the cylinder noted as α could be obtained.

$$\alpha = \arccos\left(\frac{\mathbb{P}_{cb} \cdot \mathbb{V}_c}{\|\mathbb{P}_{cb}\| \, \|\mathbb{V}_c\|}\right). \tag{3.16}$$

The magnitude of the cylinder force is the difference between the forces acting in the positive and negative directions of the hydraulic cylinder. These forces are determined by multiplying the surface area of the piston by the pressure. Hence, the forces in the cylinder $\mathbb{F}_c$ are defined as:

$$|\mathbb{F}_c| = A_{positive} \cdot p_{positive} - A_{negative} \cdot p_{negative}. \tag{3.17}$$

Using the angle α between the boom and the cylinder force vector, the effective force of the cylinder acting on the boom F_c can be determined:

$$\mathbb{F}_c = R_y(\beta)\left[\cos(\alpha)\ |\mathbb{F}_c|,\ 0,\ \sin(\alpha)\ |\mathbb{F}_c|\right]. \tag{3.18}$$

The resulting torque from the force applied by the cylinders can now be determined by:

$$T_c = \mathbb{P}_{cb} \times \mathbb{F}_c = [0, -l_{cb} sin(\alpha)|F_c|, 0]. \tag{3.19}$$

3.1.2 Estimating unknown geometric parameters

The model described in Section 3.1.1 relies significantly on geometric details such as the mass and length of the boom, the surface areas of the hydraulic cylinder, and the locations where the cylinder connects to both the machine and the boom. As this thesis aims to assess the possibility of creating a payload estimation system that does not rely on prior knowledge about the machine's geometry, the geometric parameters needed for the mechanical model would need to be estimated.

To achieve this, sensor data was collected during two separate lifting sequences with known payloads: one sequence with an empty bucket and another with a payload near the machine's maximum load capacity. This data served as the foundation for constructing a system of equations, which included the sensor readings for linkage angles and hydraulic pressures as well as the known payloads as fixed inputs and the unknown geometric parameters as the variables to be determined.

To solve for these variables, the Levenberg-Marquardt algorithm was employed, a technique for solving non-linear least squares problems. This method allowed for the indirect determination of geometric parameters from the sensor data, avoiding the need for direct geometric measurements.

3.1.3 Estimating payload using the mechanical model

Using the estimated geometric parameters, the payload mass m_l can be derived from sensor readings. Using Equation 3.1, the torque due to payload mass T_l can be isolated as follows:

$$T_l = -(T_b + T_c). \tag{3.20}$$

The force exerted by the payload is derived using the cross product of T_l and the position vector of the payload, adjusted for the angle of the boom:

$$\mathbb{F}_l = R_y(-\beta)\frac{T_l \times r_l}{\|r_l\|^2}. \tag{3.21}$$

Here, r_l is the rotated coordinate vector of the payload and $R_y(-\beta)$ is the rotation matrix for the angle $-\beta$. The payload mass can then be calculated by isolating the vertical component of the force vector, which is directly proportional to the gravitational acceleration.

3.2 Specific payload estimation model

After collecting sensor data from various machine models as described in Section 3.5, the L120H was selected as the most suitable for evaluating a payload estimation model. This choice was based on the fact that its size is relatively average compared to other machines and the extensive data available. Therefore, the development of the deep learning model started with the development of a model exclusively for the L120H.

Given the time series nature of the data, the architecture of the deep learning model was developed in such a way that it could leverage this format effectively. Thus, RNNs were found to be suitable because of their ability to understand temporal relationships. Experiments were conducted with various RNN structures, including LSTM layers, GRU layers, and combinations of convolutional layers with recurrent layers. In addition, different combinations of regularization layers were tested, such as batch normalization and dropout layers.

In order to find the best suited architecture for the model, multiple different architectures were trained and tested on the same train-test split and evaluated against each other. To optimize the hyperparameters for a given model, hyperparameter tuning was conducted with a combination of random search and hyperband tuning.

3.3 Adaptable payload estimation model

Building on the payload estimation model developed exclusively for the L120H, the next step was to extend its applicability to other wheel loader models. By training a neural network using datasets from multiple different wheel loaders, a neural network referred to as the "general model" was developed. This model was designed to fit

any machine by fine-tuning it to the specific machine where it was going to be used. The following sections will cover the architectural choices and testing for the general model as well as the fine-tuning.

3.3.1 Development of general model

The architecture of the general model originated from the best performing architectures found during the development of the specific model, as described in Section 3.2. However, for the general model, two variations were tested: one in which the number of neurons in the top layers was increased and additional dense layers were added at the bottom, and another in which the original architecture was retained but introduced a new output layer. The tested changes were done partly because of the increased capacity to learn advanced patterns due to the increased data size when training on multiple machines, and partly to enable efficient fine-tuning during the transfer learning.

The general model was developed using datasets that were collected and processed in the same way as for the specific model, this process is outlined in Section 3.5. The training data used for the general model included data from all wheel loader models from which data was available, except for the L120H, which was reserved for fine-tuning. The architecture of the general model can be seen in Figure 3.2.

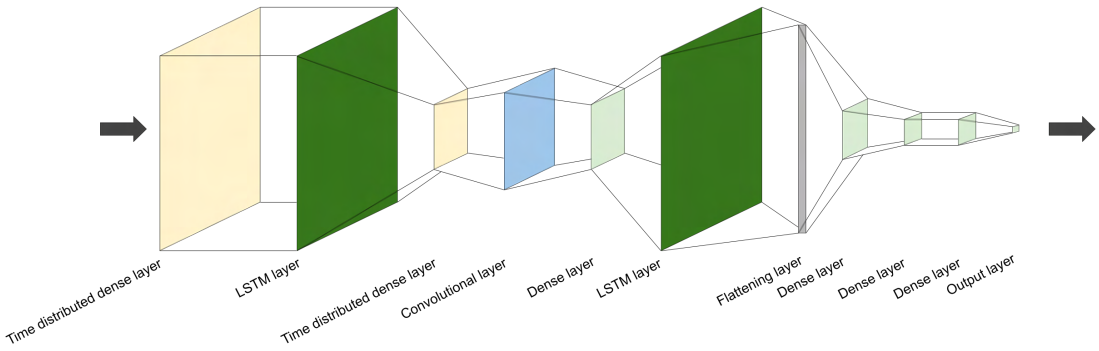


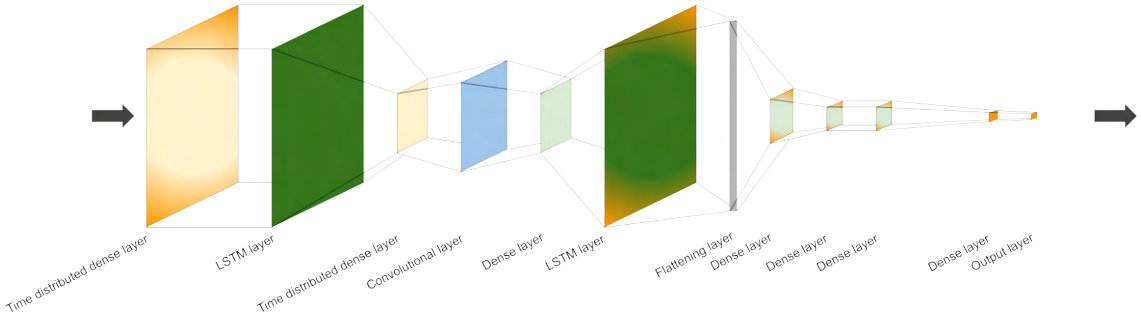
Figure 3.2: A schematic illustration showing the different layers in the general model. The arrows indicates the direction the data flowed thought the model.

3.3.2 Fine tuning of general model

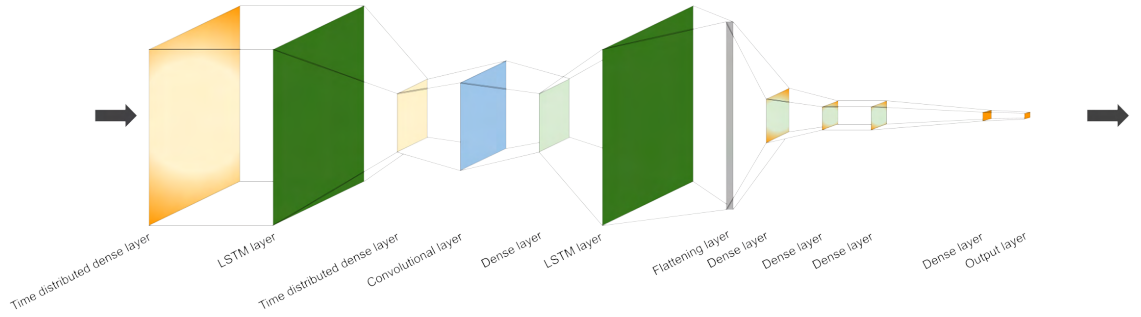
When fine tuning the general model to fit a specific wheel loader, some modifications were made to the general model. The model architecture and weights were kept intact except for the output layer, which was replaced with a 15-neuron dense layer followed by a new output layer. This reconfiguration was intended to adapt the feature mappings generated by the general model to the specific characteristics of the target machine. The weights for all layers except for the first and last four or five layers were frozen, which allows only these layers to be updated during training.

It was also tested not to freeze any layer and thus update all weights. The architecture and distribution of the fine-tuned layers can be seen in Figure 3.3.

The data used for fine-tuning came from the same Volvo L120H training set that was used for the specific model. This wheel loader model was completely unseen for the general model.



(a) Five layers fine-tuned.



(b) Four layers fine-tuned.

Figure 3.3: The architectures of the models that were fine-tuned to fit a certain wheel loader model. The orange layers in the top of the network are newly added layers and the once before are from the general model. The layers with orange edges are layers with non-frozen weights.

During development, the fine-tuned model was trained multiple times with varying data volumes to determine the optimal amount of data required for effective fine-tuning. The quantity of data required was assessed both in terms of the required number of known payloads and the required number of data samples per payload needed to achieve robust model performance. In a practical implementation, it is desired to have a low number of known payloads as well as a low number of required samples, as this data would need to be collected for each individual machine.

3.4 Classification model for sensor data reliability

Due to the nature of the operations of a wheel loader, some sensor data sequences are less reliable for the estimation of the payload. For example, when scooping up

material from a pile, external forces from the pile can distort payload measurements, and similarly, when emptying the bucket, there will be sequences where the payload in the bucket will not remain constant. Other factors, such as machine accelerations or the additional forces required to overcome static friction as the linkage transitions from static to moving states, can also decrease estimation accuracy.

To address these challenges, a new neural network, hereafter referred to as the classifier, was developed to review sensor data sequences. The classifier was designed to identify which sequences are likely to yield accurate payload estimates, these can be processed by the payload estimation model, and which sequences are unlikely to yield accurate estimates, these will be ignored. The classifier was trained using the same input data as the payload estimation model but with a target variable that represents the accuracy of the payload estimations for those sequences. Two architectural approaches were evaluated, a Convolutional Neural Network (CNN) and an AE. Transfer learning, in the same manner as for the payload estimation model, was also evaluated for the CNN architecture.

3.4.1 The convolutional neural network

The CNN was trained to predict the reliability of the input data by predicting the percentage of the maximum allowable error that the payload estimation model would likely produce with a given input sequence. This was done by generating payload estimates for previously unseen data using the payload estimation model, then calculating the prediction errors and normalizing them by dividing by the maximum allowed error. The CNN was then trained using the input sequences and normalized error percentages as the output targets. Classification was achieved by setting a threshold for predicted error percentages; this threshold was optimized to yield the highest accuracy for the binary classification.

3.4.2 The autoencoder

The AE was specifically designed to classify input sequences using anomaly detection techniques. This classifier was trained exclusively on sequences for which the payload estimation model generated estimates within the acceptable error margin. The goal of the training was to recreate the distribution of the input sequences. A sample that diverges from this distribution would be regarded as an anomaly and is therefore classified as a non-reliable sequence.

3.4.3 Adaptable classification network for sensor data reliability

An adaptable classification network was developed in order to aid the fine-tuned payload estimation model. The adaptable classification network was constructed using the same architecture as the CNN classification network. The training and

validation set for the general classifier network consisted of data from two of the wheel loader models, while the test set only consisted of data points from one of the two. The labels were calculated in the same manner as for the CNN classifier. The general classification network was then tested on the test set, both with fine-tuning and without fine-tuning of the general classifier network. This model will be referred to as the TL-CNN, as in Transfer Learning-CNN.

3.5 Data acquisition and processing

For the integration of deep learning models, it is essential to have access to a substantial volume of labeled data. Additionally, to ensure the model's effectiveness in real-world applications, it is essential to train it using as much data as possible derived from actual wheel loader operations. The available sensor suite on the wheel loader includes two hydraulic pressure sensors attached to the hoses from one of the lift cylinders, which record the hydraulic pressure on both the positive and negative sides of the cylinder. Furthermore, the machine has three IMUs, one positioned on the main body of the machine, another one on the boom, and the third on the tilt linkage, each measuring angles, angular velocity and angular acceleration. In addition to angular metrics, the body IMU also measures linear acceleration in the x-direction of the machine. The following sections will cover the collection of data as well as the pre-processing of it.

3.5.1 Data acquisition

Data were collected from eight unique wheel loader models; Volvo L60H, L90H, L120H, L150H, L180H, L220H, L260H and L350H. This involved capturing readings from the IMUs and hydraulic pressure sensors from a Controller Area Network bus (CAN bus) connected to an electronic control unit in the machine. The data were captured across various operational scenarios of the wheel loader, such as varying engine speeds, stationary lifts, lifts during movement and accelerations, lifts during varying inclinations, as well as lifts with different weight distributions for the material in the bucket. To measure the actual weight of the material in the bucket, the wheel loader was weighed on a weighbridge with a certified accuracy of ± 20 kg, both in its unloaded state and then again after loading. The material used for all logs was gravel, and all gravel came from the same pile. However, the logs were captured during different parts of the year, which may have affected the water content and therefore the density of the gravel.

Sensor readings were captured as raw CAN messages, which were decoded and converted into variables with appropriate units, resulting in structured datasets for analysis and model training. Following the decoding and structuring process, the dataset included variables such as the **roll angle**, **pitch angle**, **pitch velocity**, **pitch acceleration** and **linear acceleration** of the wheel loader, **tilt angle** of the bucket, **boom angle** along with **boom angular velocity** and **boom angular acceleration**, and the **positive pressure** and **negative pressure** within the

hydraulic cylinder. An example of a decoded dataset can be seen in Figure 3.4.

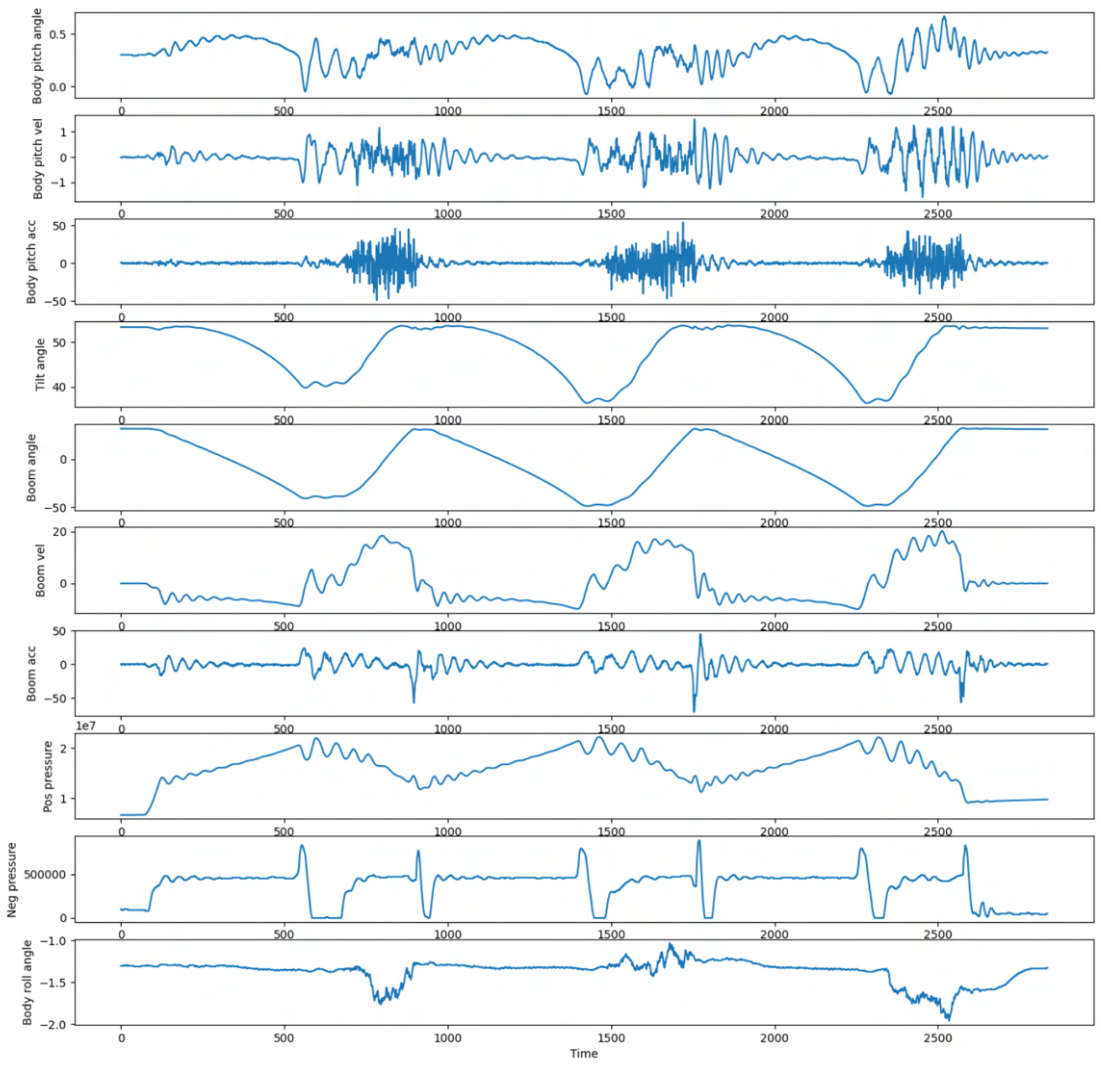


Figure 3.4: Decoded file of sensor readings from a Volvo L60H. The wheel loader was loaded with 3636 kg, and performed three lifts with idle engine speed.

3.5.2 Time-series analysis

The sensors of a wheel loader in operation will produce time-series data with a frequency of 40 Hz, creating a strong dependence between each time step due to the continuous motion of the machine. Consequently, the data were processed in continuous sequences rather than in discrete and independent time steps. In this way, the dynamics and temporal relationships within the system are utilized more effectively.

The choice of sequence length can be supported by an autocorrelation plot which shows repeating patterns in the time-series with respect to time lags. During the

wheel loader's operation, the boom oscillates due to damping effects from the BSS, creating harmonic patterns in the acceleration of the linkage components. As depicted in Figure 3.5, there is a distinct peak in autocorrelation around lag 50, suggesting that the oscillation period of the boom is approximately 50 samples. Based on this, a sequence length of 50 time steps may be a good starting point, since the autocorrelation plot displays periodic characteristics that may have predictive value. However, the sequence length chosen for the study was 20 samples. Since the auto-correlation is periodic, half the period was believed to be enough for the sequences to contain the behaviors important for the payload estimation.

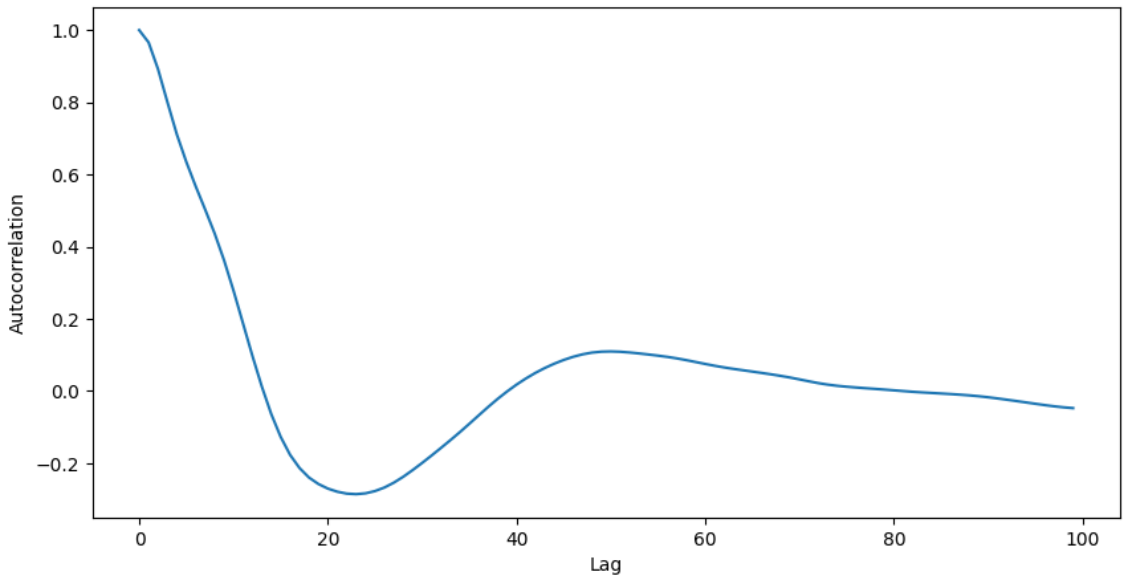


Figure 3.5: Autocorrelation of boom angular acceleration for a Volvo L60H for different lags.

A common approach to generate sequences is to use a sliding window, which creates an overlap between each consecutive sequence. This increases the amount of training data and creates a more diverse dataset [30]. To maximize data utilization, the sliding window stride was set to one. This means that each consecutive sequence overlaps with all but one of the time steps from the previous sequence.

In the development of a neural network, it is crucial to be careful when managing the data to avoid data leakage. Data leakage occurs when information meant for testing the model is used to train the model, potentially leading to overly optimistic performance estimates and poor generalization to new data. To ensure that the training, validation, and testing sets are distinct and that there is no overlap of data between them, each unique log file was put in one of the three datasets. Since each log file represents a unique sequence of data, this method segregates different operational sessions into separate sets, further minimizing the risk of data leakage.

3.5.3 Feature importance

In order to increase the accuracy of the model, an investigation of each feature's contribution to the estimated payload was made. This analysis involved two methods, correlation analysis and feature shuffling.

First, the correlation between each feature and payload mass was evaluated, with the results displayed in Table 3.1. This analysis highlighted positive hydraulic pressure as the most influential feature for payload prediction. Secondly, feature shuffling was applied, using the same network architecture as described in [7], which includes a dense layer of 24 neurons, followed by 100 LSTM cells, another dense layer of 30 neurons, and a softplus output layer. This model was trained and tested with all available features. The importance of each feature was then evaluated by shuffling one feature at a time in the test set and observing the impact on the model's Mean Absolute Error (MAE) during evaluation with the test data. Performance degradation was measured against the MAE calculated from test data with no shuffled features, as shown in Figure 3.6. The result indicates that neither **linear acceleration** nor the **pitch acceleration** contributes significantly to the estimated payload, suggesting that these could be excluded without compromising the accuracy of the estimation. Furthermore, the results indicate that the features with a high correlation with the payload, as shown in Table 3.1, also lead to significant performance degradation when removed.

Table 3.1: Correlations between the payload mass and the sensor data features. Ordered from highest to lowest correlation.

Feature	Correlation with payload mass
Payload mass	1
Positive pressure	0.6
Tilt angle	0.095
Pitch angle	0.041
Boom angle	0.039
Linear acceleration	0.006
Roll angle	-0.027
Negative pressure	-0.026
Boom velocity	-0.0018
Pitch velocity	-0.0016
Pitch acceleration	-0.0011
Boom acceleration	-0.000031

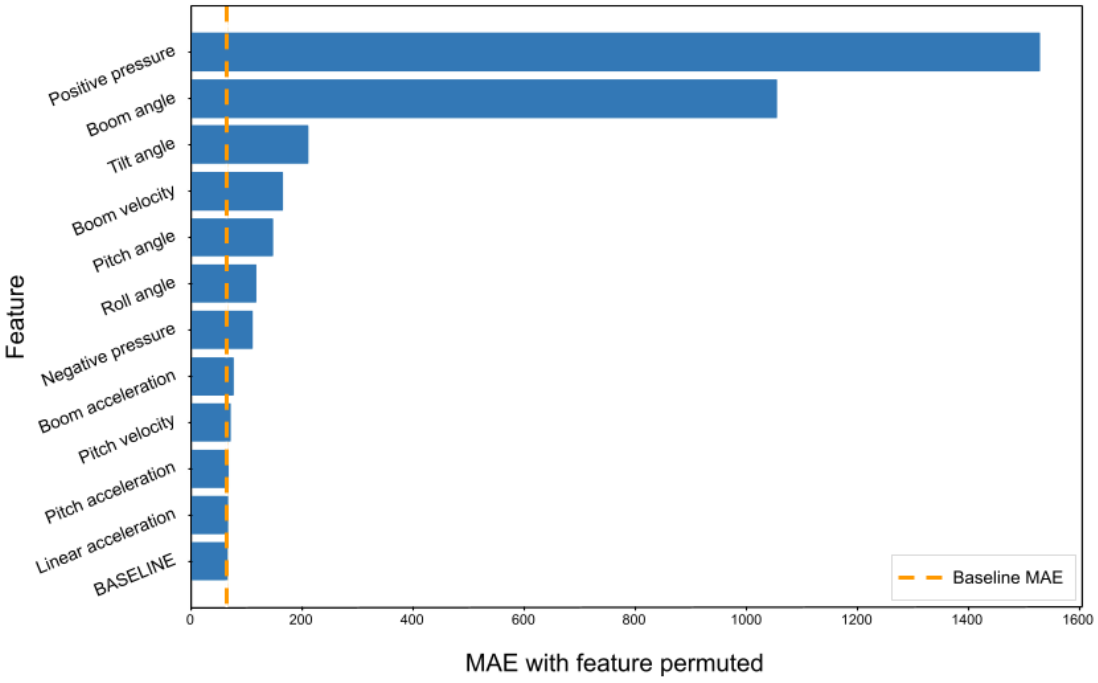


Figure 3.6: MAE for the validation of the deep learning model presented in [7], when the specified feature has been shuffled. The baseline is the MAE when non of the features was shuffled.

3.5.4 Filtering

A Kalman filter was used to refine the mechanical estimates. This was done primarily because the estimates contained occasional spikes and outliers. This step was taken as outliers can distort the neural network's interpretation of data. By reducing the influence of these outliers, the integrity of the data was maintained for model training.

3.5.5 Scaling of the data

The data was scaled before the training of the deep learning models. This was performed in order to minimize the risk of biases for the deep learning models towards different features due to the magnitudes of their values. The scaler used was the RobustScaler. It uses the median and interquartile ranges, thus reducing the impact of outliers and preserving more information in the data compared to other scalers. Thus, it is more suitable for datasets where outliers represent meaningful variations that are important for model training, rather than plain noise.

3.5.6 Data augmentation

To efficiently train a neural network, a sufficient amount of data must be available. However, since the data needed to be captured by operating wheel loaders, the accessible amount of data was limited. Therefore, data augmentation was used

to enlarge the dataset. Jittering was chosen as the data augmentation technique because it is a simple yet effective method to enhance the diversity of the dataset.

One of the early challenges in the development of the neural network model was the low number of unique labels in the dataset relative to the amount of data. This issue arose because the same payload was used to collect a large volume of data, in an effort to save time. Consequently, the neural network was unintentionally trained to classify seen labels rather than generalize to unseen data. The introduction of jittering significantly mitigated this problem by increasing the variability of the labels in the training set.

The augmentation process was carried out by initially segregating 20% of the log files to serve strictly as a test set. This step was taken to prevent data leakage and ensure that test data remained unchanged, which ensures realism for model evaluation. The remaining 80% of the dataset, designated for training and validation, was jittered using Gaussian noise with a variance of 1. The low variance was used to minimize the risk of introducing a pattern within the data that would cause the natural behavior, used by the neural networks during the payload estimation, to vanish. For each entry in the dataset, one new augmented sample was generated, which means that the volume used for training and validation was doubled as a result of this process. This augmentation not only increased the size of the dataset, but also enhanced its diversity by introducing a wider range of target weights, as all augmented samples obtained a unique payload.

Additionally, an alternative strategy was explored where the validation set was jittered separately, in contrast to jittering the training and validation set together. This approach aimed to reduce data leakage between training and validation sets by making sure that no altered artifacts of the training set contaminated the validation set. However, this method resulted in lower test accuracy, likely due to the model being exposed to less variation during training,

3.6 Evaluation

To produce results comparable with each other, evaluation metrics and criteria were established.

3.6.1 Scope of evaluation

The current payload estimation system from CPAC Systems AB requires a specific lifting motion to generate payload estimates. Given the thesis's objective of exploring whether neural networks can enhance the robustness of payload estimation systems, the evaluation will extend beyond lifting sequences to include a variety of operational scenarios. This approach aims to validate the system's performance across different operational contexts, which can potentially broaden its applicability and utility.

To ensure that the evaluation is carried out under realistic and relevant operational conditions, the analysis will exclude sequences in which the wheel loader is lowering the boom. From analyzing the data, it was concluded that the dynamics during boom lowering, where gravity assists in the descent of the load while the hydraulic system is braking, are significantly different from the dynamics when lifting or maintaining a fixed boom angle. Additionally, it can be assumed that the operator will rarely want to weigh the load during lowering since the typical practice is to keep the bucket close to the ground and only lift the weight once to dump it. To lift the load and then lower it down again would in most cases lead to unnecessary fuel consumption. Therefore, focusing on lifting sequences aligns the evaluation with the most likely use cases in real-world operations.

To exclude boom lowering events from the dataset, periods where the boom angular velocity indicates that the boom is moving downwards for one second consequently were removed. This approach avoids relying on specific boom angle thresholds, which vary between different wheel loader models, and provides a more universal solution.

3.6.2 Evaluation metrics

To assess the performance of the systems, multiple metrics were used. The first is MAE, which is the average error in a set of predictions. MAE is derived as:

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|\,. \tag{3.22}$$

Here, y_i represents the ground truth for a sample i, $\hat{y}_i$ denotes the prediction of the model for a sample i, and N is the total number of samples.

The Mean Absolute Percentage Error (MAPE) is defined as the average of the absolute differences between predicted and actual values, divided by actual values, expressed as a percentage:

$$\text{MAPE} = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{y_i - \hat{y}_i}{y_i}\right|. \tag{3.23}$$

As mentioned in Section 1.2, the goal with the project is to develop a system that can produce estimates with an error within 1% of a full-scale load. The reason for focusing on a full-scale load, rather than individual payload mass, is to standardize the percentage error metric across the entire range of payloads, in order to avoid the disproportionate penalty that MAPE imposes on smaller payloads. To quantify a full-scale load, the *static tipping load at full turn* is used. This metric represents the maximum load that the machine can lift before tipping, and is provided in the machine's technical documentation. The Absolute Percentage Error relative to the full-scale load (APE_{max}) is calculated for individual estimates according to Equation 3.24, where y_{max} represents the static tipping load at full turn:

$$\text{APE}_{\text{max}} = \frac{|y - \hat{y}|}{y_{max}}. \tag{3.24}$$

Consequently, the Mean Absolute Percentage Error relative to the maximum value ($\text{MAPE}_{\mathbf{max}}$) can be defined as follows:

$$\text{MAPE}_{\text{max}} = \frac{1}{N} \sum_{i=1}^{N} \frac{|y - \hat{y}|}{y_{max}}. \tag{3.25}$$

To further quantify performance in meeting the 1% accuracy target, we introduce the 1% Relative Error Rate (1%RER). This metric calculates the proportion of estimates within 1% of the full-scale load ($\text{APE}_{\text{max}} < 1\%$):

$$1\%\text{RER} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1} \left(\frac{|y_i - \hat{y}_i|}{y_{max}} < 0.01 \right) \tag{3.26}$$

In this equation, $\mathbb{1}$ is an indicator function that equals 1 when the condition is met and 0 otherwise.

The performance of the classifier was assessed using a confusion matrix. The primary goal of the classifier is to prevent the payload estimation model from producing inaccurate estimates. Therefore, minimizing the False Positive Rate (FPR) is crucial and was thus used as the main metric for evaluating the classifier's effectiveness. The FPR is defined as in Equation 3.27.

$$\text{FPR} = \frac{FP}{FP + TN} \tag{3.27}$$

Here, FP represents the number of false positives, and TN represents the number of true negatives.

4
Results

This chapter presents the result of the models developed.

4.1 Mechanical model

The mechanical model was evaluated on a Volvo L120H using a randomly selected test set that contained multiple different payloads. The same logs are also used to evaluate the deep learning models in the following sections. The performance metrics for the mechanical model are summarized in Table 4.1.

Table 4.1: Error metrics for mechanical model, tested on Volvo L120H test set

Metric	Value
MAE	2147.91 kg
MAPE	61.37 %
MAPE_{max}	17.18 %
1%RER	4.52 %

Figure 4.1 shows a histogram of the error distribution for the predictions of the mechanical model. Despite the high variance, the errors are centered around zero, suggesting that the model is capable of predicting the payload to a certain extent.

4.2 Specific payload estimation model

The best performing architecture for the model trained specifically for Volvo L120H was found to be a combination of LSTM, convolutional, and time-distributed dense layers. The final model achieved an estimation error within 1%, for 66% of the sequences in the test set.

The best performing architecture included a time-distributed dense layer of 600 neurons, a LSTM layer of 600 LSTM-cells, a time-distributed dense layer of 200 neurons, a time-distributed dense layer of 250 neurons, a convolutional layer of 300 filters, a kernel size of 3 and L_2 regularization, a dense layer with 200 neurons, a dense layer of 100 neurons, a dropout layer with a 30% dropout rate, a flattening layer,

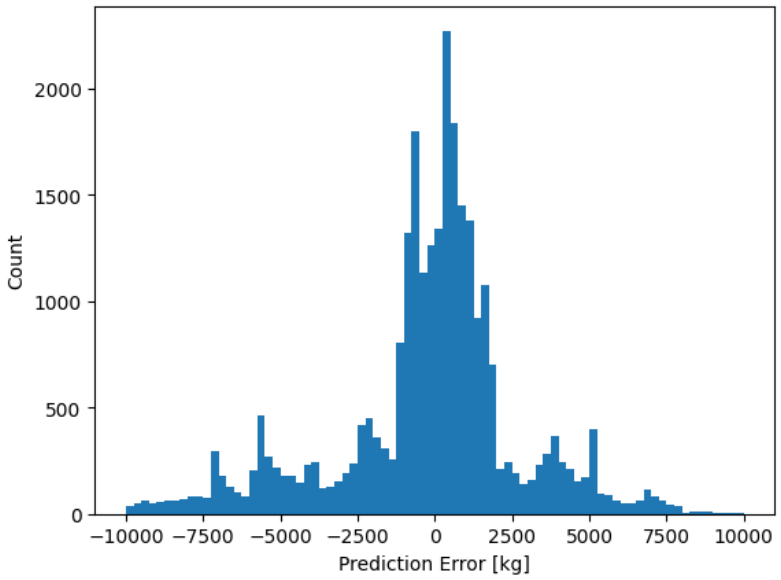


Figure 4.1: Histogram over the prediction error for the mechanical model, tested on Volvo L120H test set.

a dropout layer with 20% dropout rate, a dense layer with 150 neurons, two dense layers of 80 neurons and finally an output layer. The LSTM layer used a sigmoid activation function, and the output layer used softplus activation, while the other layers used Rectified Linear Unit (ReLU) activation. The best performance was found with a sequence length of 20 time steps per sequence. A simplified schematic diagram of this model is shown in Figure 4.2.

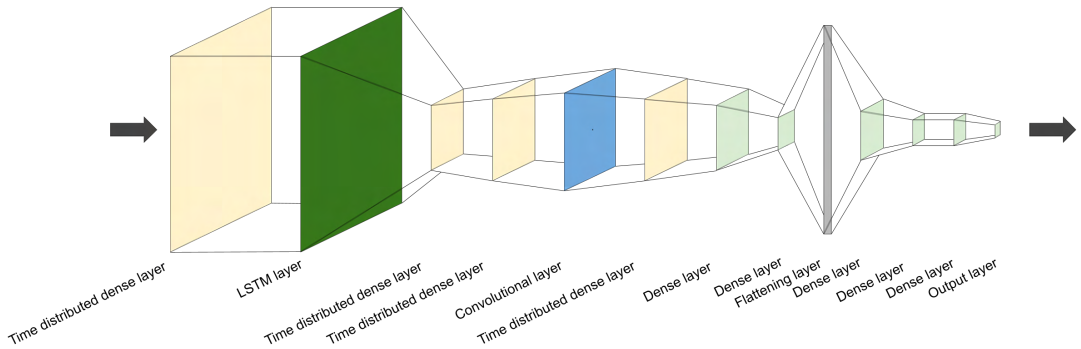


Figure 4.2: Schematic illustration over the different layers in the payload estimation model with the best performance for the wheel loader model L120H.

The model was tested using the same dataset as for the mechanical model in 4.1, its performance metrics are summarized in Table 4.2 and the histogram in Figure 4.3 illustrates the distribution of prediction errors.

Table 4.2: Error metrics for deep learning model trained specifically for Volvo L120H, tested on Volvo L120H test set.

Metric	Value
MAE	154.4 kg
MAPE	4.82 %
MAPE_{max}	1.24 %
1%RER	66.16 %

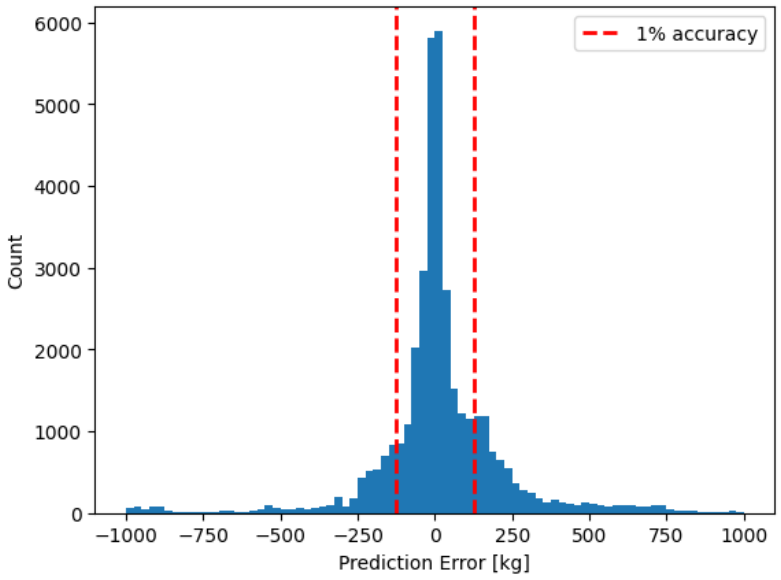


Figure 4.3: Histogram over the prediction error for the deep learning model trained specifically for Volvo L120H, tested on Volvo L120H test set. 1% accuracy (± 125kg) is shown as two vertical red lines

When comparing these results with the results of the mechanical model in Section 4.1, it is clear that the neural network-based method performs much better.

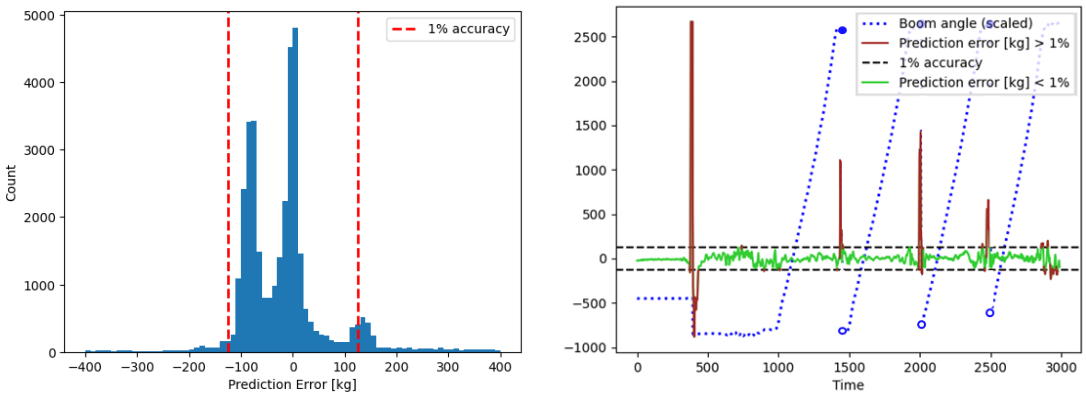
4.3 Adaptable payload estimation model

The best performing model was the general model fine-tuned across all layers with 12 distinct payloads. This configuration achieved a 1%RER of 84.45%. The evaluation metrics for this model are summarized in Table 4.3.

Furthermore, Figure 4.4a illustrates the error distribution through a histogram, while Figure 4.4b displays a time series plot of the estimated payloads compared to the boom angle for a selected log file. As observed, the model produces accurate estimates when the boom is stationary or lifting the load, and it generates outliers at the top end position.

Table 4.3: The metrics for the adaptable payload estimation model that provided the lowest MAE. The model was generated by fine tuning all layers of the general model using 12 unique payloads.

Metric	Value
MAE	111.68 kg
MAPE	3.49 %
MAPE_{max}	0.89 %
1%RER	84.45 %



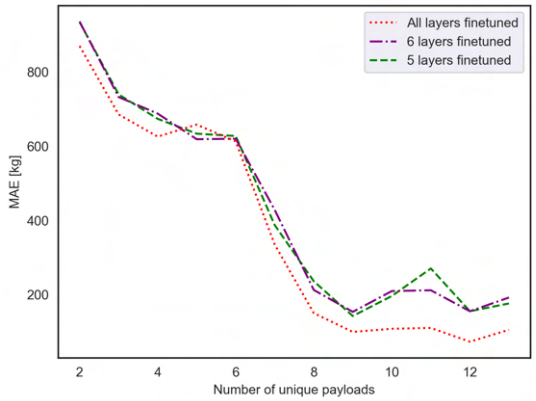
(a) Histogram over estimates for all sequences in the Volvo L120H test set.

(b) Estimates for a single log file in L120H test set over time, compared to boom angle

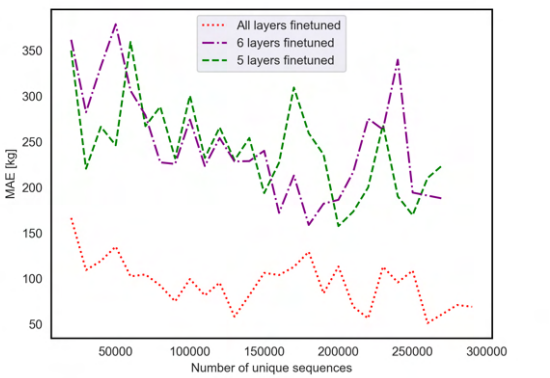
Figure 4.4: Estimated payloads for general deep learning model fine-tuned for Volvo L120H. 1% accuracy (± 125kg) is shown as dashed lines

Figure 4.5a shows the relationship between the MAE and the number of unique payloads used for fine-tuning. It also shows the impact of fine-tuning different numbers of layers. The result shows that the MAE decreased with the increase in unique payload labels in the training set. The result also indicates that the fine-tuning plateaus after around 8 unique payloads, generating MAEs between 200 and 100. The best performance was reached by fine-tuning all layers using 12 unique payloads.

Figure 4.5b shows how the number of unique sequences used for fine-tuning affects the MAE. This reveals a correlation between an increase in the number of sequences and a reduction in MAE. The lowest MAE, recorded at 51.15 kg on the validation set, was obtained when all layers were fine-tuned with a dataset comprising 260,100 unique sequences. Note that there exists data leakage between the validation set and the training set due to the data augmentation process. Consequently producing lower MAE on the validation set compared to the one generated on the test set.



(a) The MAE as a function of the number of unique payloads used for fine-tuning.



(b) The MAE as a function of the number of unique sequences used for fine-tuning.

Figure 4.5: The MAE as a function of (a) the number of unique payloads in the training set, and (b) the number of unique sequences in the training set. Note that the result in (b) are using the optimal number of unique payloads found from the result in (a). Specifically, 12 payloads for fine tuning of all layers and 9 payloads for the other two.

4.4 Classification model for sensor data reliability

The performance of the classification network varied between the three models tested. The results for the CNN's suggest that they were able to identify patterns linking the input data to the payload estimation model's ability to achieve predictions within the 1% accuracy goal, while the AE result did not suggest the same. Using the confusion matrices of the classification models, seen in Figure 4.7b for CNN, Figure 4.9b for the AE and Figure 4.10a for the TL-CNN, the FPR and accuracy could be calculated. These metrics are presented in Table 4.4 for the different classification models.

Table 4.4: The FPR and accuracy for the three different models developed for classification of the sensor data reliability.

Model	FPR [%]	Accuracy [%]
CNN	58.68	71.66
AE	94.73	67.42
TL-CNN	45.03	73.14

The precision of the CNN indicates that a pattern was found between sensor data and reliability. However, the FPR indicates that more than half of the unreliable data were classified incorrectly, suggesting flaws in the found pattern. In Figure 4.7a,

predictions made by the CNN is presented. The results show that the distribution of false positives is mostly consisting of data points producing errors relatively close to the allowed error threshold.

The CNN that proved to generate the highest accuracy in combination with lowest FPR, consisted of 11 layers, a time-distributed dense layer of 64 neurons, followed by a convolutional layer with 64 filters of kernel size 3, and a dropout layer set at 40%, a time-distributed dense layer of 48 neurons, a convolutional layer with 48 filters, a dropout layer of 40%, a time-distributed dense layer with 32 neurons, a convolutional layer with 32 filters and kernel size 3, a dropout layer of 40%, a flattening layer, and an output layer. All layers had ReLU activation except the output layer that had softplus. The architecture can be seen in Figure 4.6.

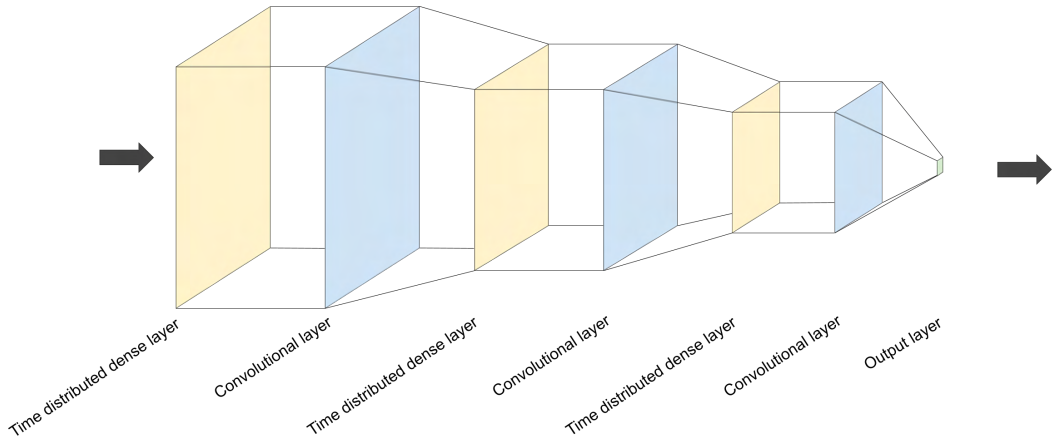
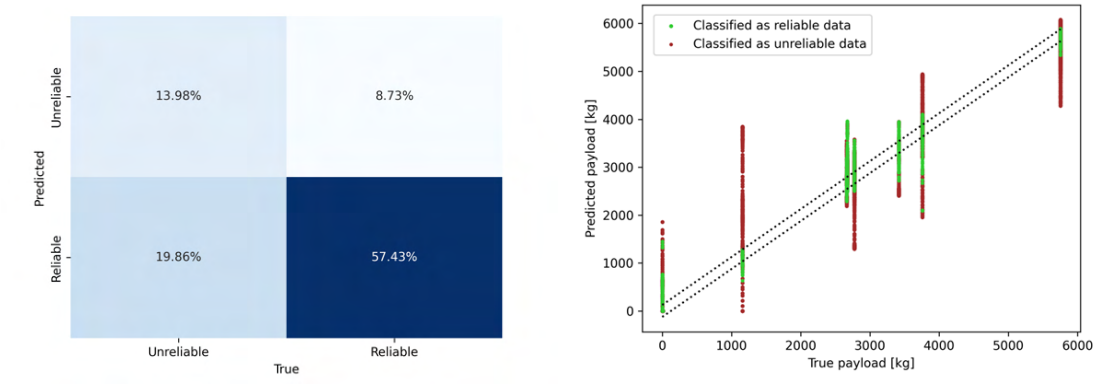


Figure 4.6: A schematic illustration of the best performing architecture for the CNN classifier.

The performance of the AE model demonstrates a significant bias towards classifying sequences as reliable. As shown in the confusion matrix in Figure 4.9b, the AE classified a substantial majority of the samples as reliable. This is also reflected in the FPR, which is derived in Equation 3.27. For the AE, the FPR was found to be 94.73 %, which shows that the model is unable to identify unreliable data in the vast majority of cases.

The AE that proved to generate the highest accuracy in combination with lowest FPR, was built using dense layers. The encoder consisted of three dense layers with 16, 8 and 4 neurons and a dropout layer with a dropout percentage of 20% between them. The decoder was the reverse of the encoder, three dense layers with 8, 16 and 200 neurons with the same type of dropout layers in between. All layers used ReLU activation. The architecture can be seen in Figure 4.8.



(a) Confusion matrix. **(b)** Classification.

Figure 4.7: Classification performed by the CNN. (a) present the confusion matrix, and (b) present a visualisation of the classifications of the different sequences and the payload estimation they yielded.

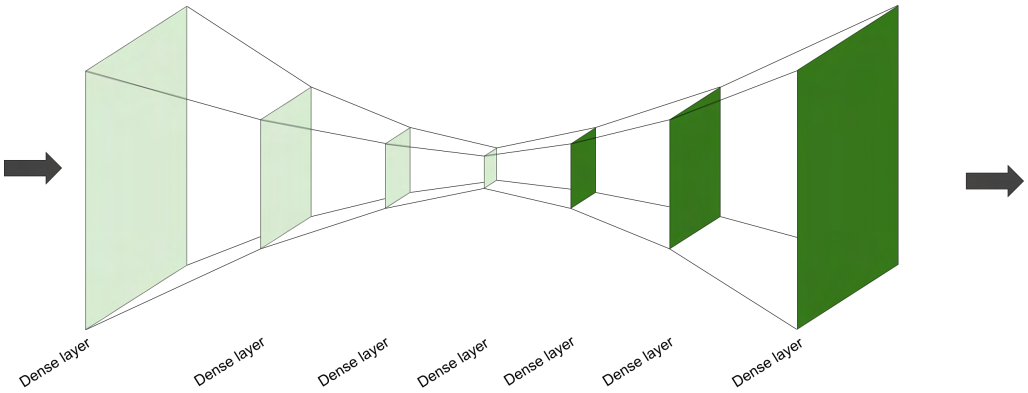
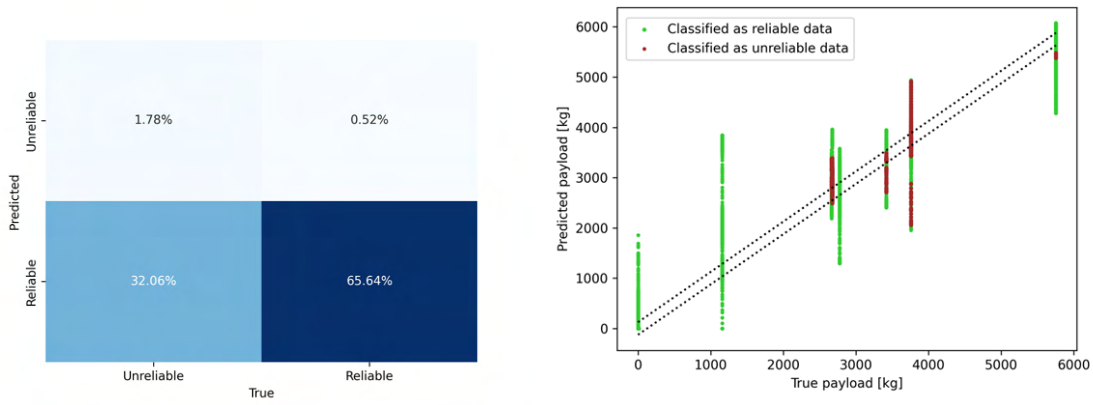


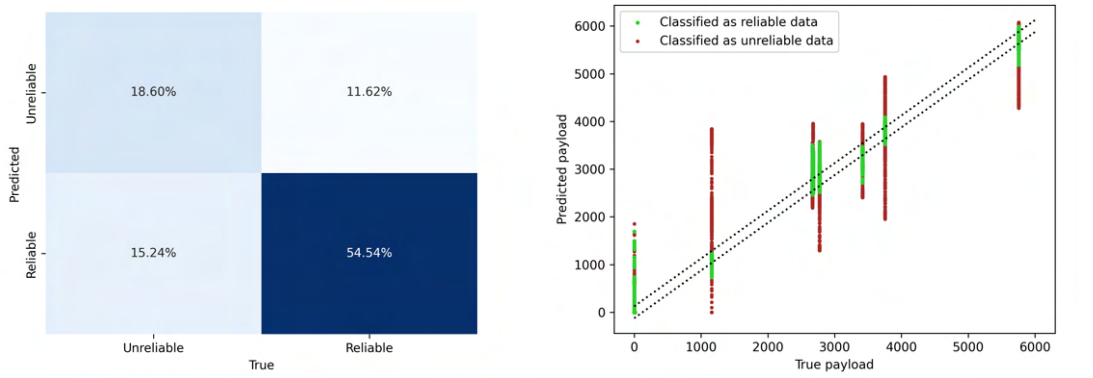
Figure 4.8: A schematic illustration of the best performing architecture for the AE classifier.

The TL-CNN provided an FPR of 45.03% and an accuracy of 73.14%. Compared to the CNN, the adaptive classification model proved to be more accurate. A visualization of the classifications made by the adaptable classification model can be seen in Figure 4.10b. The results show that the distribution of false positives is mostly consisting of data points producing errors relatively close to the allowed error threshold. However, for the payload 0 kg the spread is much more diverse compared to the CNN, seen in Figure 4.7a.



(a) Confusion matrix. **(b)** Classification.

Figure 4.9: Classification performed by the AE. (a) present the confusion matrix, and (b) present a visualisation of the classifications of the different sequences and the payload estimation they yielded.



(a) Confusion matrix. **(b)** Classification.

Figure 4.10: Classification preformed by the TL-CNN. (a) present the confusion matrix, and (b) present a visualisation of the classifications of the different sequences and the payload estimation they yielded.

5
Discussion

The results imply that it is possible to use deep learning models to achieve payload estimations within the a 1% accuracy target. However, the results are also somewhat ambiguous, which is the topic that will be discussed in the following paragraphs.

5.1 Mechanical model

The results of the mechanical model demonstrate its ability to estimate the payload mass to a certain degree, which is notable because it does not have any knowledge of the machine's geometry. To our knowledge, achieving such results without geometric data has not been documented in research before. However, the robustness of the model is notably low, resulting in a high variance in the error. These findings suggest that developing a universally adaptable and robust payload estimation model solely based on analytical estimates, without detailed geometric information, may not be feasible.

The simplicity of the mechanical model used in this thesis could be a factor in its limited performance. The model presumes ideal conditions and does not take into account factors such as friction or hydraulic pressure drops, implying that a more sophisticated mechanical model incorporating additional variables might yield improved results.

Despite these limitations, the current model shows potential as an input to deep learning models. The primary objective of integrating a mechanical model into a neural network is to reduce the problem's dimensionality. Provided that the estimates reflect the true distribution of the data, it can be assumed that they can benefit the neural network. However, minimizing the error in the mechanical model is advantageous as it reduces the complexity required of the neural network to achieve accurate predictions.

5.2 Specific payload estimation model

The deep learning model exclusively developed for the Volvo L120H shows a major improvement over the mechanical model for the same test data. This demonstrates the potential of neural networks for payload estimation tasks. The results are promising, with an MAPE_{max} of 1.24% and a 1%RER of 66.16 %. These metrics show considerable progress compared to the MAPE_{max} of 17.18% and a 1%RER of

4.52 % achieved by the mechanical model. Despite these improvements, the model does not meet the target of a MAPE_{max} under 1 %. However, it is important to note that the model was tested in challenging operational scenarios. Unlike the existing system developed by CPAC Systems AB, which only produces payload estimates during lifting sequences, this model was subjected to a wide range of operational conditions. This underscores the robustness of the neural network, showing its ability to accurately estimate payloads even in challenging scenarios.

Although these results are promising, the practical application of this model is limited to the specific Volvo L120H machine from which the training data has been collected. As outlined in the research discussed in Section 1.1, collecting extensive data from each individual machine is not feasible, and a more universally applicable solution that requires less specific training data is required to industrialize the system on a large scale.

5.3 Adaptable payload estimation model

The adaptable payload estimation model proved to reach a higher 1%RER in comparison to the model only trained on a single wheel loader model. The result also shows that the adaptive payload estimation model required a relatively small quantity of data to fine-tune relative to training a model from scratch. However, the results also highlighted that increasing the number of payloads and the quantity of training data generally led to a decrease in MAE. This presents a trade-off between estimation accuracy and the quantity of data used for fine-tuning. This aligns with our expectations, as a broader data distribution allows the model to generalize better.

When fine-tuning the general model, the amount of layers used for fine-tuning proved to be essential. Fine-tuning all layers proved to almost always result in a lower MAE than using only a few layers. The results also indicated that using six layers to fine-tune often produced a lower or the same MAE as fine-tuning five layers. This indicates that the fine-tuned model benefited from having as many layers as possible fine-tuned. This finding was surprising, since most research has found that fine-tuning only the outer layers is beneficial. This may indicate that the architecture of the general model is too compact to capture the full complexity of the features. This in turn may suggest that all layers needed to be fine-tuned in order to effectively specialize the model for specific models.

5.4 Classification model for sensor data reliability

The three models, CNN, AE and TL-CNN, tested for classification of the reliability of the data showed varying results. The CNNs displayed the ability to predict the reliability of the data with an accuracy greater than 70%. The AE on the other hand proved to be inadequate in predicting the reliability of the data. One possible

reason for the different outcomes is that the models were built to comprehend the correlations within the time series differently. The AE were taught to recreate the input data, generating the classification of the data from the magnitude of the reconstruction error. Since the AE was constructed with only dense layers, there is no evidence suggesting that the model could have learned the temporal relationships in the data. The CNNs on the contrary were trained to predict the payload estimation models' prediction error, producing predictions based on the largest error allowed for the payload estimation model. As the name implies, the CNNs were built by convolutional layers, making it possible for the model to learn temporal relationships.

Another feasible explanation is based on the amount of data used to train the different models. In order to achieve anomaly detection with the use of an AE, the AE needs to be trained on the system's ordinary behavior. In the case of this study, the ordinary behavior refers to the situation in which the payload estimation model is able to produce an estimation within the allowed error margin. The optimal payload estimation model found produced predictions within the margin in approximately 66% of the operational time (excluding lowering events). This generated a training set for the AE with almost half the number of data points as the training set for the CNN, which includes all points independent of the predictions of the payload estimation models. This may have caused the AE not to have enough data to properly understand the ordinary behavior of the system, resulting in large reconstruction errors for the ordinary behavior as well. Therefore, a high threshold error was used for the classification of the data. This theory may be supported by the CNN result. Although the result indicates that the model was able to classify the sequences, 30% of the cases were incorrectly classified. This might be a consequence of a complicated task, yet it can also be the fallout from a limited training set. The improved performance of the TL-CNN suggests that a larger dataset might decrease misclassifications.

The result indicated that there exists a connection between the behavior of the input data and the payload estimation models' estimation error because of the result from the CNNs. Since the AE did not show an indication of finding this pattern, it may imply that the relationships resulting in a higher estimation error by the payload estimation model lie inside the temporal changes within the sequences. If this is the case, reconfiguring the AE with an architecture capable of understanding these temporal relationships, such as convolutional layers or LSTM layers, might enhance its classification accuracy. However, due to the success of the CNN, this was not investigated.

In order to achieve an adaptable classification network, the same approach was used as for the adaptable payload estimation models. A general model was developed and tested both with and without fine-tuning. However, in order to receive the data and labels to use to generate a general classification model, there needed to be access to fine-tuned payload estimation models. That is because the prediction error acquired from those models was the label for the classification model. The generation of fine-tuned models requires a large proportion of the data, leaving only

a small amount for training, validation, and testing of the general and fine-tuned classification network. Furthermore, due to the project's time limitations, only two fine-tuned payload estimation models were able to be generated in time for the development of the adaptable classification network. This resulted in a small dataset for the general model and its fine-tuning.

The result for the TL-CNN compared to the CNN shows a decrease in FPR and a small increase in accuracy. A feasible explanation for this is the availability of more diverse data. When data is received from different machines during training, the distribution of different operational scenarios increases. Consequently, the classification network has the opportunity to better understand the behavior that causes failures in the payload estimation. However, the small increase in accuracy compared to the larger decrease in the FPR indicates an increase in the False Negative Rate (FNR). The increase in FNR may be explained by the larger dataset. Since the dataset consisted of data from two different wheel loader models, with two different 1% accuracy thresholds, ambiguity may have been introduced within the labels. Consequently, increasing the FNR.

5.5 Data properties

The data utilized for developing both the payload estimation models and classification networks were collected from wheel loaders operating at a designated test site. This location was used because of the presence of a certified weighbridge needed to measure the payload accurately. All data were collected with the same gravel material and under a predetermined set of operational scenarios. This may hinder generalizability for the model when applied to other operational environments or materials with significantly different properties, such as mud or large rocks. Such conditions could introduce unfamiliar patterns in the input data, which may result in prediction errors. Expanding the diversity of operational tasks and materials in the training data could enhance the model's robustness but would likely also increase the data requirements for effective fine-tuning.

Similarly, the machines used for data collection used the same attachment throughout the entire data collection, only one unique machine of each machine model was used, and all machines used were made from the same manufacturer. Given these constraints, it is not possible to assert with certainty that the results obtained are universally applicable to wheel loaders of different manufacturers. However, the adaptable classification network and the adaptable payload estimation model have shown promising signs of applicability across various models of Volvo wheel loaders.

The splitting of data into three sets, training, test, and validation, was done with the purpose of preventing data leakage, as detailed in Section 3.5. This process involved a random selection of log files and assigning them to one of the three datasets before transforming these files into sequences used for training. This comes with a risk of unevenly distributing complex data scenarios, such as sequences with abrupt

braking, high acceleration, or high engine speeds, across the datasets. Consequently, this could lead to biased results if the test set consists predominantly of simpler operational scenarios, from which it is easier for the model to estimate the payload accurately.

During the initial processing of the data, the feature importance assessment indicated that neither the linear acceleration nor the pitch acceleration of the wheel loader had any effect on the precision of the payload predictions of the used deep learning model. Regarding the pitch acceleration, this was not very surprising, as the BSS can be assumed to prevent the pitch accelerations to propagate to the linkage. However, for the linear acceleration it was more unexpected, given that the forces acting on the boom are influenced by the vehicle's linear acceleration. A possible explanation for this result is that the deep learning model might be able to infer linear acceleration from the temporal relationships in the angular data of the boom. It is important to note that the feature importance measures were determined early on in the project with another architecture than what is used in the final models. Thus, while there is a chance that including linear acceleration or pitch acceleration could enhance the accuracy of the final payload estimation model, its earlier redundancy in the simplified model suggests that this improvement might be minimal.

Figure 4.4 shows the prediction errors against the boom angle over time, showing that the model reliably estimates payloads except for sequences near boom angle discontinuities. The discontinuities arise from the exclusion of all events in which the boom is lowered, as detailed in Section 3.6.1. The sequences used for training and testing were generated in a manner that ensures that a sequence would never overlap one of these discontinuities. Therefore, the discontinuities themselves do not directly cause unreliable payload estimates. However, sequences close to these excluded events affect the model's performance, potentially due to residual behaviors from the lowering of the boom that are not entirely removed. This suggests that the method used to exclude lowering events might need refinement. A good alternative would be to not include sequences with a boom angle outside a certain range, close to the extremities of the boom angle, in contrast to the current solution which excludes lowering events based on the angular velocity of the boom.

5.6 Future work

The result presented in this thesis allows further research to be carried out in the field of payload estimation for construction equipment using deep learning, as well as industrial implementations. However, in order to produce a better understanding within both fields, more research is crucial. Firstly, this study did not explore the performance of the proposed models in operating wheel loaders; it was limited to offline analysis using log files. Further work should aim to apply these models in real-time scenarios to better understand their industrial applicability and performance implications.

Although the results of the thesis suggest that payload estimation using an LSTM-based neural network is feasible, further investigation of the optimal architecture for a payload estimation model could advance the field. In this thesis, only RNN-based architectures have been evaluated. However, there are many other architectures that could be interesting to test, such as transformers and CNNs. Additionally, since the adaptive payload estimation model was developed using an architecture similar to the best one identified for the specific payload estimation model, a new adaptive payload estimation model developed using the newly optimized architecture may also yield better performance.

The results also suggest that transfer learning is effective for the task, even yielding better results with less specific data compared to the model trained specifically for a single machine. The general model was trained on data from wheel loaders of varying sizes, and the L120H which the general model was later fine-tuned to was not included in the general model's dataset. Nevertheless, the size of this wheel loader was within the range of sizes present in the dataset of the general model. This raises the question whether these results can be generalized to wheel loaders of all sizes, and more specifically, to wheel loaders that are larger or smaller than the ones in the dataset for the general model. To answer this question, further investigation is necessary into the adaptability of the general model. Another question is whether this general model can produce reliable payload estimations when fine-tuned for other types of construction equipment, such as excavators. This would also benefit from similar research.

Considering the limitations of the dataset discussed in Section 5.5, the study could not definitively determine whether the payload estimation model would perform similarly with materials other than gravel. To address this, more data should be collected and used that encompass a variety of materials and operational conditions. This would help to assess the model's adaptability and validate its utility across different operational contexts.

The classification network proved to be able to produce predictions of data reliability with an accuracy of 73%. However, since the classification did not work perfectly, both reliable and unreliable sequences were removed and not fed to the payload estimation models. Consequently, generating a lower 1%RER for the payload estimations than the adaptive payload estimation model alone. To achieve a higher 1%RER, alternative approaches to classify reliable and unreliable sequences should be explored, for example, limit the operational scenarios in which payload prediction can be performed. A more detailed analysis would then be necessary to determine where the payload estimation model performs well and where it does not. As mentioned earlier, another approach for removing the lowering events might be beneficial. Another strategy may be to further explore the use of anomaly detection, yet with the capacity to understand temporal behaviors within the data sequences.

The presented solution proposes a general model trained offline that is fine-tuned online during the setup of the machine. However, with the use of reinforcement

learning, the fine-tuning of the models could be carried out continuously during the regular operation of the wheel loader. The problem that needs to be solved for such a solution to work is how the sensor data should be labeled. One idea of how this could be done would be to provide feedback to the model based on how the pitch of the machine behaves. When the wheel loader is lifting the load up and down, it is common for it to rock back and forth, sometimes so much that the rear tires lift from the ground. This means that there exists a torque around the front axle and, therefore, a force equilibrium for the wheel loader not to tip over. Using pitch acceleration and possibly also tire pressure, one may be able to estimate the weight of the payload or at least provide feedback to a reinforcement learning model. However, due to time constraints, this idea was not examined during this thesis.

The payload estimation produced in this thesis was done with the help of a deep neural network. This resulted in no understanding of how the network concluded the estimated weight of the payload. For example, there is no plausible way to conclude how the network established that the linear acceleration of the wheel loader did not contribute to the estimation of the payload. Since the problem of payload estimation has a clear mechanical setup, there exist strong reasons to believe that it would be plausible to develop a more interpretable payload estimation model. A more interpretable model would further aid the understanding of the payload estimation problem, since the models chain of thought may reveal knowledge about the problem not investigated.

6
Conclusion

This study aimed to examine whether deep learning technologies could be utilized to create a universally applicable payload estimation system for wheel loaders. The main objective was to investigate how accurate payload estimation could be achieved with limited or no information about the specific geometry of the machine. To fulfill this, a system based on recurrent neural networks and transfer learning techniques has been developed.

The results demonstrate the potential of deep learning models for this purpose. The developed system was able to provide payload estimates with an error within 1% of a full-scale load for approximately 84% of the operational time, excluding periods when the wheel loader is lowering the boom. A contributing feature in this system is a simplified and universally applicable mechanical model that is able to adapt itself to a specific machine using data from two different lifts, one with an empty bucket and the other with nearly full load. Based on this data, all the necessary geometric parameters of the wheel loader could be approximated. Although the mechanical model alone did not produce reliable estimates, it was helpful as a feature to improve the performance of the subsequent deep learning model.

The deep learning model, together with the transfer learning technique, proved to result in a payload estimation model able to successfully adapt to various wheel loader models. This was done with fine-tuning, which proved to require less data from a specific machine compared to developing a tailored deep learning model for it. Suggesting that this may be a favorable approach if data availability is constrained for a specific wheel loader. However, the use of transfer learning requires a substantial initial dataset comprising diverse operational data from multiple wheel loaders. The results also highlighted a notable trade-off between the accuracy of payload estimations and the volume of data used for fine-tuning.

In conclusion, the combination of estimated geometric parameters and transfer learning allowed the deep learning model to achieve 1% precision in the estimation of payload for 84% of the operational time under study, without prior knowledge of the specific characteristics of the wheel loader and with less data than would be needed to develop a dedicated model for each machine. Although promising, more research is required to implement these strategies in real-time operations of wheel loaders. However, this thesis is believed to have proven that the concept works with confident results.

Bibliography

[1] A. Renner, H. Wind, and O. Sawodny, "Online payload estimation for hydraulically actuated manipulators," *Mechatronics*, vol. 66, p. 102 322, Apr. 2020, ISSN: 0957-4158. DOI: 10.1016/J.MECHATRONICS.2020.102322.

[2] A. Walawalkar, S. Heep, M. Frank, R. Leifeld, and C. Schindler, "Validation of an analytical method for payload estimation in excavators," Jan. 2018.

[3] M. Savia and H. Koivo, "ON-LINE PAYLOAD DETERMINATION OF A MOVING LOADER USING NEURAL NETWORKS," *IFAC Proceedings Volumes*, vol. 35, no. 1, pp. 37–42, Jan. 2002, ISSN: 1474-6670. DOI: 10.3182/20020721-6-ES-1901.01157.

[4] J. y. Feng, W. Chen, T. j. Wang, P. Tan, and C. s. Li, "Online prediction of loader payload based on a multi-stage progressive model," *Automation in Construction*, vol. 144, p. 104 621, Dec. 2022, ISSN: 0926-5805. DOI: 10.1016/J.AUTCON.2022.104621.

[5] J. Hindman, R. Burton, and G. Schoenau, "An Artificial Neural Network Approach to Payload Estimation in Four Wheel Drive Loaders," in *ASME International Mechanical Engineering Congress and Exposition, Proceedings*, vol. 9, Oct. 2007. DOI: 10.1115/IMECE2007-41618.

[6] A. Walawalkar, S. Heep, M. Frank, and C. Schindler, "Payload Estimation in Excavators Using a Machine Learning Based Parameter Identification Method," in *Lecture Notes in Mechanical Engineering*, Springer Science and Business Media Deutschland GmbH, 2020, pp. 1670–1680, ISBN: 9783030380762. DOI: 10.1007/978-3-030-38077-9{_}190.

[7] G. Chris, D. Greff, M. Starke, and M. Geimer, "Development and evaluation of a weighing system for forestry cranes based on artificial neural networks," *Landtechnik*, vol. 74, pp. 102–115, Oct. 2019. DOI: 10.15150/lt.2019.3213.

[8] M. Wahde, *Biologically inspired optimization methods: an introduction*. WIT press, 2008, ISBN: 1845641485.

[9] R. C. Staudemeyer and E. R. Morris, "Understanding LSTM – a tutorial into Long Short-Term Memory Recurrent Neural Networks," Sep. 2019. [Online]. Available: http://arxiv.org/abs/1909.09586.

[10] H. Salehinejad, S. Sankar, J. Barfett, E. Colak, and S. Valaee, "Recent Advances in Recurrent Neural Networks," Dec. 2017. [Online]. Available: http://arxiv.org/abs/1801.01078.

[11] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, ISSN: 08997667. DOI: 10.1162/neco.1997.9.8.1735.

[12] Tufts University and IEEE Circuits and Systems Society, *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS) : August 6-9, 2017, Boston, MA, USA*, ISBN: 9781509063895.

[13] D. M. Hawkins, *Identification of Outliers*. Springer Netherlands, 1980. DOI: 10.1007/978-94-015-3994-4.

[14] M. Ahmed, A. Naser Mahmood, and J. Hu, "A survey of network anomaly detection techniques," *Journal of Network and Computer Applications*, vol. 60, pp. 19–31, Jan. 2016, ISSN: 1084-8045. DOI: 10.1016/J.JNCA.2015.11.016.

[15] V. T. Le and Y. G. Kim, "Attention-based residual autoencoder for video anomaly detection," *Applied Intelligence*, vol. 53, no. 3, pp. 3240–3254, Feb. 2023, ISSN: 15737497. DOI: 10.1007/s10489-022-03613-1.

[16] S. Alam, S. K. Sonbhadra, S. Agarwal, and P. Nagabhushan, "One-class support vector classifiers: A survey," *Knowledge-Based Systems*, vol. 196, p. 105 754, May 2020, ISSN: 0950-7051. DOI: 10.1016/J.KNOSYS.2020.105754.

[17] R. Nawaratne, D. Alahakoon, D. De Silva, and X. Yu, "Spatiotemporal anomaly detection using deep learning for real-time video surveillance," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 1, pp. 393–402, Jan. 2020, ISSN: 19410050. DOI: 10.1109/TII.2019.2938527.

[18] T. Li, M. L. Comer, E. J. Delp, *et al.*, "Anomaly Scoring for Prediction-Based Anomaly Detection in Time Series," Tech. Rep.

[19] G. Iglesias, E. Talavera, Á. González-Prieto, A. Mozo, and S. Gómez-Canaval, *Data Augmentation techniques in time series domain: a survey and taxonomy*, May 2023. DOI: 10.1007/s00521-023-08459-3.

[20] T. T. Um, F. M. Pfister, D. Pichler, *et al.*, "Data augmentation of wearable sensor data for Parkinson's disease monitoring using convolutional neural networks," in *ICMI 2017 - Proceedings of the 19th ACM International Conference on Multimodal Interaction*, vol. 2017-January, Association for Computing Machinery, Inc, Nov. 2017, pp. 216–220, ISBN: 9781450355438. DOI: 10.1145/3136755.3136817.

[21] A. Nikitin, L. Iannucci, and S. Kaski, "TSGM: A Flexible Framework for Generative Modeling of Synthetic Time Series," May 2023. [Online]. Available: http://arxiv.org/abs/2305.11567.

[22] H. Alqahtani, M. Kavakli-Thorne, and G. Kumar, "Applications of Generative Adversarial Networks (GANs): An Updated Review," *Archives of Computational Methods in Engineering*, vol. 28, no. 2, pp. 525–552, Mar. 2021, ISSN: 18861784. DOI: 10.1007/s11831-019-09388-y.

[23] F. Zhuang, Z. Qi, K. Duan, *et al.*, *A Comprehensive Survey on Transfer Learning*, Jan. 2021. DOI: 10.1109/JPROC.2020.3004555.

[24] S. J. Pan and Q. Yang, *A survey on transfer learning*, 2010. DOI: 10.1109/TKDE.2009.191.

[25] P. Gupta, P. Malhotra, L. Vig, and G. Shroff, "Transfer Learning for Clinical Time Series Analysis using Recurrent Neural Networks," Jul. 2018. [Online]. Available: http://arxiv.org/abs/1807.01705.

[26] G. Vrbančič and V. Podgorelec, "Transfer learning with adaptive fine-tuning," *IEEE Access*, vol. 8, pp. 196 197–196 211, 2020, ISSN: 21693536. DOI: 10.1109/ACCESS.2020.3034343.

[27] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" Tech. Rep.

[28] G. Napadow, "Energy Study of Bucket Positioning Systems on Wheel Loaders-Loader Linkages," Tech. Rep.

[29] A. Ayoub, C. M. Berg, N. Xiong, and A. Papadopoulos, "DESIGN OF A DYNAMIC BOOM SUSPENSION SYSTEM IN A HYBRID WHEEL LOADER," Tech. Rep., 2018.

[30] A. Girma, X. Yan, and A. Homaifar, "Driver Identification Based on Vehicle Telematics Data using LSTM-Recurrent Neural Network," Apr. 2019. DOI: 10.1109/ICTAI.2019.00127.

DEPARTMENT OF MECHANICS AND MARITIME SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY