Field test of neural-network based automatic bucket-filling algorithm for wheel-loaders

Siddharth Dadhich\textsuperscript{a}, Fredrik Sandin\textsuperscript{a}, Ulf Bodin\textsuperscript{a}, Ulf Andersson\textsuperscript{a}, Torbjörn Martinsson\textsuperscript{b}

\textsuperscript{a}Department of Computer Science, Electrical and Space Engineering, Luleå University of Technology, 97187, Luleå, Sweden.
\textsuperscript{b}Volvo CE, Bolindervägen 5, 63185, Eskilstuna, Sweden.

Abstract

Automation of earth-moving industries (construction, mining and quarry) require automatic bucket-filling algorithms for efficient operation of front-end loaders. Autonomous bucket-filling is an open problem since three decades due to difficulties in developing useful earth models (soil, gravel and rock) for automatic control. Operators make use of vision, sound and vestibular feedback to perform the bucket-filling operation with high productivity and fuel efficiency. In this paper, field experiments with a small time-delayed neural network (TDNN) implemented in the bucket control-loop of a Volvo L180H front-end loader filling medium coarse gravel are presented. The total delay time parameter of the TDNN is found to be an important hyperparameter due to the variable delay present in the hydraulics of the wheel-loader. The TDNN network successfully performs the bucket-filling operation after an initial period (100 examples) of imitation learning from an expert operator. The demonstrated solution show only 26\% longer bucket-filling time, an improvement over manual tele-operation performance.

Keywords: Neural-network, Bucket-filling, Wheel-loader, Automation, Construction

Email addresses: siddharth.dadhich@ltu.se (Siddharth Dadhich), fredrik.sandin@ltu.se (Fredrik Sandin), ulf.bodin@ltu.se (Ulf Bodin), ulf.andersson@ltu.se (Ulf Andersson), torbjorn.martinsson@volvo.com (Torbjörn Martinsson)
1. Introduction

Wheel-loaders are multi-purpose machines fitted with different kinds of attachments such as buckets and forks. In construction industry, wheel-loaders are mostly used with a bucket to transport materials such as soil, gravel and rock. Due to the lack of useful models of earth (soil, gravel or rock) for real-time control, automatic bucket filling has been an open problem for three decades [1].

Tele-operation of earth-moving machines is commercially available for mining industry [2] and is being researched for construction application [3–5]. Tele-operation is considered as a step towards fully automated machines, but it results in reduced productivity and fuel efficiency [6–8]. Incorporating driver assistant functions such as automatic bucket-filling can improve the usefulness of tele-operation by enabling one remote-operator to control multiple machines. Tele-remote operation, without automatic bucket-filling, results in 70% longer cycle time of operation and a productivity loss of 42% compared with manual operation [8]. Driver assistant functions can potentially improve performance when operating machines on tele-remote, especially the bucket-filling operation which is difficult to perform with the constrained perception of the remote operator [8].

Fig. 1 shows our experimental wheel-loader, a Volvo L180H machine, with an operator performing bucket-filling on medium-coarse gravel. Operators make use of vision, sound and vestibular feedback to perform the bucket-filling operation with high productivity and fuel efficiency. To avoid both translation and lift stalls, the expert operators make intermittent use of the tilt action when moving the machine forward, and lifting the bucket upwards. Expert operators also prevent wheel-spin during bucket-filling by a combination of lift and tilt actions or by reducing the applied torque by lowering the throttle.

Research aiming to automate earth-moving machines has a long history [9–11] but a commercial system with autonomous earth-moving machines has not been demonstrated [12]. However, recently Dobson et al. [13] presented a solution for autonomous loading of fragmented rock for load-haul-dump (LHD) machines using an admittance controller actuating the tilt piston only, i.e., the bucket-filling is performed using only the curl movement of the bucket. LHD machines have (1) longer buckets and (2) higher breakout forces (the maximum amount of force the machine generates while curling the bucket) compared to wheel-loaders and are specifically build this way
for underground mining applications. Dobson et al. [13] reports 61% less bucket-filling time and 39% increase in the bucket-weight compared to a single manual operator. However, the control scheme presented in [13] cannot be used for wheel-loaders because they have lower breakout force which require both lift and tilt pistons to be actuated simultaneously to complete the bucket-filling operation.

Automatic bucket-filling via model-based control is a challenging problem because it has proven difficult to develop a good model of the interaction forces between the bucket and pile [14]. Classical control theory has been applied to this problem in the form of trajectory control [15], compliance control [16] and feed-forward control [12] without clear success.

An automated digging control system (ADCS) for a wheel-loader based on finite-state-machine and fuzzy-logic was demonstrated by Lever [17] on a range of rock-loading tasks. Wu [18] simulated trajectory control in rock piles with fuzzy logic and used neural-networks to model wheel-loader components. Most of the previous work require accurate models of the machine and therefore are susceptible to breakdown in the presence of modeling errors, wear and changing conditions.

We take a machine learning approach to automate the bucket-filling operation for front-end loaders and focus here on medium-coarse gravel. This approach is motivated by the difficulties in modeling the material to be loaded.
and in estimating the interaction forces between the material and the bucket.

Since an inexperienced human can learn the bucket-filling task with some practice, in particular with homogeneous materials, we consider the possibility to develop an end-to-end machine learning approach to this problem. Initially, the aim was to predict the control actions (joystick signals) of a wheel-loader operator as the bucket moves through the pile. The prediction of control actions is a time-series regression problem, which also appears in many other contexts such as prediction of rainfall [19] and energy consumption of buildings [20].

Artificial neural networks (ANN) or simply neural networks are models capable of learning behavior of complex systems from data. Different types of neural networks have been used for financial time-series prediction, such as time-delayed network [21], ensemble of networks [22], convolution network [23] and recurrent network [24]. Neural networks have also been used in other time-series prediction application such as battery state estimation [25], prediction of energy consumption [26] and the control of HVAC systems [27].

In former work we found that a linear regression model is unable to capture the on-off nature of joystick commands, but the bucket-filling problem is tangible with machine learning and a multi-layer classifier for lift/tilt actions [28]. Several machine learning models for prediction of the control actions of wheel-loader operators are investigated in Dadhich et al. [29] and it was concluded that a relatively simple three-layer time-delayed neural network (TDNN) outperforms several other machine learning models, such as regression-trees.

In this work, we modify a regular feed-forward TDNN so that the hidden layer implements a softmax function and is exposed to multi-level categorical outputs (6 classes each for lift/tilt) during training. From an implementation point of view, our architecture is inspired by the concept of regression by classification [30].

A convolution neural network (CNN) with a fixed-size 1-D convolution layer at the input is equivalent to the TDNN [31]. Both TDNN and 1-D CNN use a finite-length context window to learn spatio-temporal patterns in the data. Alternatively, recurrent neural networks (RNN) with feedback connections resulting in an infinite-length context window are used to model sequential data. In this work, the choice to use a shallow TDNN is motivated by the limited size of training data (limited duration of imitation learning accepted) and simplicity. Training of deep RNNs is more challenging due to problems with exploding/vanishing gradients [32] and the function of the
resulting network is more complex.

In this paper, we present, analyze and demonstrate a solution based on a
time-delayed neural network (TDNN) and imitation learning for autonomous
bucket-filling in a wheel-loader. The proposed method is not designed for a
particular machine or pile. Thus, in principle, the method involving the
use of TDNN and imitation learning can be used in other machines and on
different pile environments.

The main contributions of this paper are (1) presenting an TDNN-based
solution for bucket-filling, (2) studying the effect of different design-choices
and hyperparameters on the bucket-filling performance and, (3) presenting a
comparison of the performance of the automatic bucket-filling algorithm and
an expert operator.

2. Time delayed neural network

Artificial neural-networks (ANNs) are bio-inspired models and computa-
tional systems consisting of interconnected elements called neurons (or units)
[33]. These neurons are typically arranged in layers, thus ANNs typically have
a layered architecture. The most simple ANNs have three layers, an input
layer (sensors), a hidden layer (processing) and an output layer (actuators),
and a feed-forward architecture where information flows from input to output
in a sequential way. A three layer feed-forward ANN is shown in Fig. 2. A
single hidden-layer neural network is a universal function approximator, i.e.,
it can compute any continuous function [34] with an appropriate selection of
neuron parameters. Thus, ANNs are useful in a wide range of applications
where the relationship between inputs and outputs is complex and has to be
fitted to data.

The time-delayed neural-network (TDNN) was first introduced by Waibel
et al. [35] for phoneme recognition in speech. TDNNs are useful when the
information about the input-output relationship is spread across time and
input signals. The TDNN architecture enables the network to discover fea-
tures and temporal relationships between features independent of position in
time [35]. In a TDNN, each neuron has connections to every neuron in the
previous layer and also to past (time-delayed) activations of the neurons in
the previous layer. A three layer TDNN is shown in Fig. 3, which includes
time delays at the input layer only. The delay step, $DS = n t_s$, and the to-
tal delay time, $TDL = k n t_s$, are hyperparameters of the input layer of the
TDNN. In Fig. 3, with $n = 4$, $k = 2$, $t_s = 50$ ms and $p = 1$, implies that the
Figure 2: A three-layer artificial neural network with $p$ input features, $m$ hidden units and $q$ outputs.

Figure 3: A three-layer time-delayed neural network with $p$ input features, $x_1,...,p$, step size $n$, sampling time $t_s$, delay step $nt_s$ and total delay time $knt_s$. The network has $m$ hidden units and $q$ outputs. This TDNN architecture includes time-delays at the input layer. The total number of parameters (neuron weights and biases), $n_P$, is $p(k + 1)m + mq + m + q$. 

6
input layer has three features, which are $x_1$, $x_1(t - 200\, ms)$ and $x_1(t - 400\, ms)$. Furthermore, if $m = 4$, i.e., there are four hidden units, then there are a total of 12 connections between the input layer and the hidden layer.

Each neuron in the hidden and output layers of the TDNN in Fig. 3 include an activation function, which depends on the cumulative sum of all incoming inputs weighted by connection weights, $w_c$, for each connection, $c$, between two units. Furthermore, each neuron $n$ of this TDNN has a bias, $b_n$, which is independent of the input connections and is added to the weighted cumulative sum. The outputs of the hidden layer in Fig. 3 are

$$a_h = \phi_h \left( \sum_{i=0}^{k} \sum_{j=0}^{p} w_{hij} x_j(t - ni) + b_{1h} \right), \quad (1)$$

and the outputs of the output layer are

$$y_l = \phi_l \left( \sum_{h=0}^{m} w_{lh} a_h + b_{2l} \right), \quad (2)$$

for each unit $h$ and $l$, respectively. The activation functions, $\phi_h$ and $\phi_l$, are in general non-linear functions. The total number of parameters (weights and biases), $n_P$, for the TDNN in Fig. 3 is $p(k + 1)m + mq + m + q$.

Commonly used activation functions in ANNs include log-sigmoid, tan-sigmoid, rectifier and the softmax function. These non-linear functions enable the network to capture non-linearities in the input-output relationship. A comparison of different activation functions in [36] argues that a rectifier function lead to better results in many applications and is more plausible in a biological perspective than log/tan-sigmoid functions. A softmax activation function is commonly used on outputs of all neurons of the last layer of neural networks performing classification tasks. Softmax normalizes an array, $A$, of real numbers to another array $A_{sm}$ of real numbers of the same dimension as $A$, so that $a_i \epsilon [0, 1]$ and $\sum a_i = 1 \forall a_i \epsilon A_{sm}$. Thus, the softmax function creates a probability distribution from an array of output values. Therefore, the softmax function is useful for decision making networks that need to select one outcome among several possible outcomes.

There are several different types of ANN that can be used to model input-output relationships that are spread across time and input signals, such as recurrent ANN. The high cost of producing training samples is our main motivation for considering a TDNN model with relatively few parameters.
and low model complexity. As shown below, a TDNN model is sufficient to obtain a functioning solution for the type of material considered here after a short period of training by an expert operator.

3. Methodology

3.1. Experiment setup

The Volvo L180H wheel loader, shown in Fig. 1, is equipped with sensors to record the pressures in the lift and tilt hydraulic cylinders. The machine is modified to read and write signals on the Canbus connected to the machine ECU (electronic control units). This gives the possibility to record internal signals like the engine RPM, and the position and velocity of the lift and tilt joints. The speed of the machine can be estimated from the sensor on the drive axle, which measures angular velocity. However, we directly use the drive-axle angular speed as one of the input features to the neural network model.

The bucket linkage of the Volvo L180H wheel-loader is depicted in Fig. 4, showing the location of lift and tilt angle encoders. From Fig. 4, the lift angle is defined as the angle between the machine’s horizontal to the OE link (the boom). The tilt angle is the angle between the OE link and the GDF link (the tilt lever). The angle sensors are absolute encoders with 0.12° resolution.

The wheel-loader is also equipped with a load-weighing system which shows the weight in the bucket with ±1% error. The load-weighing system uses the lift pressure sensors and data from three IMUs, mounted one each on the boom, the front-frame and the rear-frame, to estimate the weight in the bucket. The final weight is obtained when the bucket is being lifted after the end of bucket-filling.

The data from the machine, such as the drive-axle angular speed, engine rpm, gear, steering, throttle, lift/tilt cylinder angles, angular velocities and the joystick signals applied by the operator are logged. The joystick outputs between 0–5 volts (2.5 V at neutral position). The range used for actuation of pistons is 0.7–2.3 V (extension) and 2.7–4.3 V (retraction) while 2.3–2.7 V is considered as deadzone, to prevent unintended use. Since bucket-filling with wheel-loader on medium coarse gravel involves only the extension of pistons, the range 0.7–2.3 V is the only useful part of joystick signal. While using the joystick signal for training the neural-network, we have range-normalized it from 2.3–0.7 to 0–1 range, where one represents maximum velocity demand.
The data from the pressure transducers in the lift/tilt cylinders is used to calculate the net force applied by the hydraulics on the lift/tilt pistons, $F_{\text{piston}} = A_C P_C - A_R P_R$, where $A_C$, $A_R$ and $P_C$, $P_R$ are the areas and measured pressures on the cylinder and rod side of the piston, respectively. The data is logged at 50 Hz and the signals from the lift/tilt cylinders and drive-axle speeds are filtered with a 60 ms (three time steps) moving-average filter.

The material used in the experiment is medium coarse gravel with fine particles up to 64 mm in diameter. This material is not as difficult to scoop as blasted rock, but more complex than fine gravel and sand. In total, 96 bucket-filling examples are recorded with an expert operator, who is one of the best at Volvo’s test facility in Eskilstuna, as found out in [37]. The data is collected in a controlled manner, i.e. the operator is instructed to maintain an engine speed of 1300 RPM (~50% throttle), to obtain maximum power from the engine.

During data collection, the operator performs the bucket-filling operation and then lifts the bucket to measure the weight. The material is unloaded at the same place as it was loaded. This procedure leads to a variation of the shape and slope of the material in the pile. Therefore, many bucket fillings are needed to discover general scooping patterns that applies to different pile...
shapes. However, the slope of the pile is maintained at about 30–35° for each scooping, thus providing some control over the experimental conditions.

3.2. The different phases of the scooping operation

Our review of the recorded data and discussions with wheel-loader operators reveal that the bucket filling process is separated into four distinct phases. Before the start of phase one, the bottom of the bucket should be aligned with the horizontal plane defined by the contacts between the wheels and the ground. The bucket-filling algorithm then implements the four phases and the transitions between them as depicted in Fig. 5 and described below. The most interesting is the phase three, where the neural network operates. The rest of the phases are pre-determined after analyzing the manual operation data.

1. Approach: The throttle in phase one is 45%, which is sufficient to maintain a speed of about 3 km/h when approaching the pile. The next phase starts when the pressure in the lift cylinder rises above 80 bar, which occurs due to an internal control loop trying to compensate for the forces from the pile in order to keep the bucket in the same initial position.

2. Lift: The algorithm starts lifting the bucket with 40% lift action in order to achieve sufficient pressure on the front-wheel tires to avoid wheel-spin. This strategy is used by all operators. The next phase starts when the lift cylinder pressure exceeds 120 bar.

3. Bucket filling: The control of lift and tilt actions are determined with the artificial neural network. During this phase, a constant throttle value is used. The next phase starts when the tilt angle exceeds 105°.

4. Exit the pile: The last phase is needed to exit the pile and finish the bucket-filling process. A lift command with 40% actuation is send until the lift angle becomes zero, i.e., when the lift arm is parallel to the horizontal plane.

When the bucket-filling algorithm terminates the driver resumes control to weigh the bucket, unload and restart the bucket-filling experiment.

3.3. Lift and translation stall

A lift stall is defined when the lift command is not close to zero, i.e., their is an intend to do lifting but the lifting speed of the boom is close to zero.
Figure 5: Bucket-filling phases (a) The four phases in the bucket-filling algorithm are (1) Approach towards the pile (2) Lift with no tilt (3) Bucket-filling with neural-network and (4) Exit the pile. (b) Top: The pressure in the lift cylinder determine the switch between the first to second phase ($P > 80$ bar) and the second to the third phase ($P > 120$ bar). Middle: The joystick signals during the manual operation indicate the on-off use of tilt action. Bottom: Phase 3 ends when the tilt angle exceeds $105^\circ$. 
Figure 6: Lift and translation stall during one bucket fill. The high value of lift action (middle) and low value of lift velocity indicate a lift stall (top). Low values of the drive-axle speed (bottom) albeit the high value of throttle (middle) indicate translation stalls.
Similarly, a translation stall is defined when the machine’s forward speed approaches zero but the throttle pedal is pressed.

Fig. 6 shows an example of lift and translation stalls during one bucket-filling by the operator. The operators through experience learn to perform the bucket filling with minimal lift and translation stalls. If the machine stall frequently, it can be felt uncomfortable while sitting in the machine.

### 3.4. Performance metrics

The bucket weight (in tons), measured by the load weighing system, and the time spent in phase three (in seconds) are the performance metrics used to evaluate the performance of the bucket-filling algorithm. A full bucket of the experimental wheel-loader with medium course material weighs \( \sim 7.2 \) tons with a recommended 105% bucket-fill factor. However, many operators go for 110% bucket fill-factor which weighs \( \sim 7.6 \) tons. The expert operators take between 6-8 seconds for a typical bucket-fill with our experimental setup.

### 3.5. Wheel spin

Wheel-spin is the undesirable event when the applied forces on the wheel exceeds the available friction force, resulting in a loss of traction. Wheel spin damages tires and results in significant increases in the operational cost [38]. The problem to measure and avoid wheel-spin is difficult and has been studied in [6]. In this work, we do not focus on estimating or compensating wheel-spin, which is both an interesting and challenging problem. In our experiments, in order to avoid wheel-spin, we use moderate values of the throttle during phases one to three (40–55%). However, for more difficult materials such as very coarse gravel and rock, higher values of throttle throughout the bucket-filling process may be required.

### 3.6. Neural-network

Motivated by the results of former studies [28, 29] two network architectures are considered for the automatic bucket-filling algorithm. The regression model (Fig. 7a) is a TDNN with one hidden layer, as shown in Fig. 3. This network produces output signals in each time step and is trained using the mean-squared-error between the predicted and targeted lift and tilt signals from the operator training dataset.

The architecture of the classification model is motivated by the observed behavior of the expert operator. It can be noted in Fig. 6 (middle) that the lift and tilt signals appear to be used by the operator at different levels (high,
Figure 7: Two time-delayed neural network architectures that have been trained for bucket-filling. The middle layer and last layer are fully connected in both networks. (a) The regression architecture is the simple three-layer neural network with one hidden layer with 12 units \((m = 12)\). (b) The classification architecture implements the middle layer as a softmax layer for lift/tilt joystick outputs that has one neuron for each of six classes for both lift and tilt. The twelve neurons are fully connected to the two output neurons.
<table>
<thead>
<tr>
<th>Class</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$j_S &lt; 0.1$</td>
</tr>
<tr>
<td>2</td>
<td>$0.1 \leq j_S &lt; 0.3$</td>
</tr>
<tr>
<td>3</td>
<td>$0.3 \leq j_S &lt; 0.5$</td>
</tr>
<tr>
<td>4</td>
<td>$0.5 \leq j_S &lt; 0.7$</td>
</tr>
<tr>
<td>5</td>
<td>$0.7 \leq j_S &lt; 0.9$</td>
</tr>
<tr>
<td>6</td>
<td>$j_S \geq 0.9$</td>
</tr>
</tbody>
</table>

Table 1: Definitions of the six classes of lift/tilt joystick actions used for training the middle layer of the classification TDNN model. The symbol $j_S$ represents the normalized joystick signal for both lift and tilt.

medium, low), in particular the tilt signal. This behavior is a consequence of the fact that it is difficult for a human operator to smoothly modulate two joysticks simultaneously while observing the pile, modulating the throttle and focusing on the sounds and vibrations from the machine. We mimic this multi-level joystick behavior of the operator with the classification model (Fig. 7b). For this purpose, the normalized lift and tilt joystick signals have been discretized into six classes (levels) as shown in Table. 1.

In the classification architecture, instead of a regular hidden layer, the middle layer implements a classifier that predicts one of six classes for each of the lift and tilt joystick actions. The top six neurons in the middle layer of Fig. 7b output soft-values for the lift classes, while the bottom six neurons output soft-values for the tilt classes.

The input data is range-normalized with “mapminmax” function, to have all inputs in the range $[-1, 1]$. The middle layer implements a $tansig$ function (Eq. 3) in the regression model and a $softmax$ function (Eq. 4) in the classification model. The output layer in both models implements a rectified linear unit, $ReLU(x) = \max(0, x)$.

$$tansig(x) = \frac{2}{1 + e^{-2x}} - 1$$  \hspace{1cm} (3)$$

$$softmax(x_i) = \frac{e^{x_i}}{\sum_{i=1}^{m} e^{x_i}}$$  \hspace{1cm} (4)$$

The two networks are trained with resilient backpropagation (Rprop) algorithm [39] minimizing mean-squared-error (MSE). Rprop is a gradient based optimization with self tuning step size. It is a fast, robust and memory efficient variant of the backpropagation algorithm [40]. In the classification
Figure 8: Simulation result of one test example from the middle layer of the classification model.

Figure 9: Simulation result of one test example from the output layer of the classification model.
model, the cost function also includes the class outputs from the middle layer. To avoid overfitting, we use L2-regularization of the TDNN weights.

We use cross-validation [41], which does not require splitting the data into training, validation and test sets and makes efficient use of the available data to estimate the test error. In $k$-fold cross-validation, the data is divided in approximately equal sized $k$ sets of which one set is left-out for testing and $k - 1$ are used for training. By shifting the left-out testing set, $k$ models are trained and the test error is estimated by averaging the error committed by each of the $k$-models on the corresponding left-out training set. In this paper, we use cross-validation to study the effect of different hyperparameters, such as the TDL, on the difference between the operator actions and the model predictions.

Both TDNN models are trained with 96 bucket fillings by the expert operator (imitation learning). Fig. 8 and 9 show one test example output from the trained classification model. The TDNN model captures the trends in the output signal with some delay. This delay in the simulated output is expected as there is an inherent delay in the hydraulics of the machine, due to which the features (angles/velocity/force of lift/tilt pistons) trail the output (control actions).

3.7. Model deployment

We used MathWorks environment to develop and deploy the bucket-filling algorithm. Matlab's neural-network toolbox has an implementation of TDNN, which has been used to write and train the neural networks. A real-time PC (Speedgoat), compatible with Simulink Real-Time operating system, is used to run the bucket-filling algorithm in the wheel-loader. The real-time PC connect to pressure sensors and to the ECUs (Engine control units), via the CanBus protocol.

The real-time PC has an Intel Celeron 1.5 GHz processor with 4 GB RAM. The base model is executed at 1 kHz while the neural network model runs at 50 Hz. The deployed program has an average task execution time of 58 $\mu$s.

The TDNN models, when deployed in the machine, produce noisy outputs during the bucket filling process. A low pass infinite-impulse-response post-processing filter is used to smoothen the signals sent to the machine. The filter, shown in Eq. 5, is designed for a smooth time response without introducing large time delays.
Figure 10: Low-pass filtering of the neural network output.

\[ H(z) = \frac{1 + 2z^{-1} + 1z^{-2}}{1 + 1.511z^{-1} - 0.609z^{-2}} \]  

An example of the output produced by the neural network in a field test and the corresponding filtered output is shown in Fig. 10.

4. Experimental results and analysis

A series of experiments are conducted to make the different design choices using a one-factor-at-a-time approach, and to compare the automatic-bucket filling algorithm with the expert operator. To select one of the two TDNN architectures in Fig. 7 and determine the parameters, such as the input delay and the throttle, we compare their performance one to one. The tests in sections 4.1–4.4 are performed with six test trials \( (N = 6) \), while the test in section 4.5 is performed with twenty trials \( (N = 20) \). The experiments are costly to conduct and involves driving the wheel-loader to a test facility each time, which motivates the limited number of trials in the experiments.

4.1. Classification vs regression model

The aim of this experiment is to evaluate the performance of the regression and classification architectures. We ran six trials for each model type and present the results in terms of the performance metrics in Table 2. The regression model is two times slower than the classification model. Upon investigating the signals produced by the regression model, we observe that the regression model produces smaller lift/tilt actions, resulting in longer lift
Table 2: Comparison of regression and classification models with six trials ($N = 6$). The neural-network hyperparameters for both models are $TDL = 160$ ms, $DS = 20$ ms, which gives $\sim 800$ parameters ($n_p$). In this experiment 45% throttle is used.

<table>
<thead>
<tr>
<th>Model type</th>
<th>Weight (tons)</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regression</td>
<td>7.48±0.19</td>
<td>23.70±3.43</td>
</tr>
<tr>
<td>Classification</td>
<td>7.45±0.24</td>
<td>11.44±0.48</td>
</tr>
</tbody>
</table>

and translation stalls. This is because the regression model averages different output signals. The classification model captures the multi-level behavior of control actions and therefore manages to manipulate the lift/tilt joystick to navigate with less lift and translation stalls. In the subsequent experiments, presented in sections 4.2–4.5, only the classification model is used.

4.2. Training

All models, presented in this paper, are trained with 96 bucket-filling examples by an expert operator. We find that the classification model does not function when training is carried out with 32 or 64 examples. The neural network does not produce sufficiently high values of lift/tilt action in phase three and the bucket freezes in a continuous lift and translation stall. Thus, we conclude that about 100 bucket-filling examples are sufficient to generate an operational bucket-filling neural-network model for this particular machine and material.

We investigate if the random initialization of neural-network weights and the training protocol plays a role in how the network performs. In this study, three models of the same type (classification, $TDL = 160$ ms, $DS = 20$ ms) are trained and evaluated. An evaluation with Welch’s T-test [42] show statistically significant differences between the three models for the bucket-filling time ($p < 0.01$, all combinations). The results are presented in Table A1 in the Appendix. We conclude that the random initialization results in slightly different trained networks, with small differences in the corresponding performance. Although the cost function is minimized in the training of the network, we think that this (undesirable) effect can be avoided by training with more data.

All inputs do not necessarily contribute equally in the trained network. The middle-layer weights of the trained networks can be analyzed to investigate if this is the case. Fig. 11 shows the relative importance of the input features and the delayed features in terms of the middle layer weights. It can
Lift Tilt

<table>
<thead>
<tr>
<th></th>
<th>Lift Force</th>
<th>1.0544±0.026</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lift Force</td>
<td>0.8979±0.0291</td>
<td></td>
</tr>
<tr>
<td>Lift Angle</td>
<td>0.4094±0.0374</td>
<td>0.4227±0.0389</td>
</tr>
<tr>
<td>Lift Velocity</td>
<td>0.3884±0.0342</td>
<td>0.4225±0.0322</td>
</tr>
<tr>
<td>Tilt Force</td>
<td>0.301±0.0338</td>
<td>0.3321±0.0273</td>
</tr>
<tr>
<td>Tilt Angle</td>
<td>0.3329±0.0315</td>
<td>0.3457±0.0254</td>
</tr>
<tr>
<td>Tilt Velocity</td>
<td>0.3654±0.0483</td>
<td>0.3424±0.0284</td>
</tr>
<tr>
<td>Machine Speed</td>
<td>0.3561±0.0242</td>
<td>0.4194±0.0368</td>
</tr>
</tbody>
</table>

Table 3: Weights of middle-layer connections affecting the lift and tilt outputs.

be observed in Fig. 11 that the lift pressure at time $t-160$ ms and $t-400$ ms consistently are the most significant input features in all trained networks. Some other features, such as the lift force at $t, t-240$ ms, lift angle at $t-240$ ms, $t-640$ ms and the lift velocity at $t-80$ ms and $t-640$ ms are also consistently significant. The weaker connections tend to vary between the different trained networks.

Alternatively, the root-mean-square (RMS) value of the weight vector for each input feature obtained by concatenating the delay dimension of the middle layer weight matrix provide some insight into the importance of individual features. Following the design of the middle layer of the classification TDNN model, the weights connecting the top six neurons in the middle layer are used to calculate the connection strengths for lift. Similarly, weights connecting to the bottom six neurons affect the tilt. Table 3 show the RMS values when the same model (L640) is trained 10 times. From this analysis, it is clear that the lift force is the most important feature but none of the other features appears to be insignificant.

4.3. Throttle

The recorded data with the expert operators, in uncontrolled trails, reveal that they make aggressive use of throttle when filling a bucket. However, in our algorithm, the throttle is kept constant in line with the design principles of a wheel-loader and the operator guidelines for correct bucket-filling behavior.

To evaluate the role of the throttle in the bucket-filling process with our TDNN solution, a few different throttle levels are investigated during bucket-filling (phase 3) and the results are reported in Appendix in Table A2. It is
Figure 11: Average of the positive weights from the twelve neurons in the hidden layer of the L640 model, trained four times with different initial weights. The vertical axis of each plot shows input features of the neural-network, whereas the horizontal axis shows the delayed inputs, where each step is 4 units (= 80 ms). A dark shade of a pixel corresponds to a higher value (black=1, white=0). Pixels with values higher than 0.3 are highlighted with red squares to distinguish the strong and weak connections. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)
<table>
<thead>
<tr>
<th>Name</th>
<th>TDL/(ms)</th>
<th>DS/(ms)</th>
<th>No. of parameters ($n_P$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>S160</td>
<td>160</td>
<td>40</td>
<td>~500</td>
</tr>
<tr>
<td>L160</td>
<td>160</td>
<td>20</td>
<td>~800</td>
</tr>
<tr>
<td>S320</td>
<td>320</td>
<td>80</td>
<td>~500</td>
</tr>
<tr>
<td>L320</td>
<td>320</td>
<td>40</td>
<td>~800</td>
</tr>
<tr>
<td>S640</td>
<td>640</td>
<td>160</td>
<td>~500</td>
</tr>
<tr>
<td>L640</td>
<td>640</td>
<td>80</td>
<td>~800</td>
</tr>
</tbody>
</table>

Table 4: Description of models with different delay configurations in the input layer.

observed that for a higher value of the throttle ($\approx 55\%$), the bucket weight increases at the cost of longer filling time.

4.4. Input-layer delay

The inherent dynamics of a wheel-loader from the motion of the joysticks to the bucket movement is complex, in particular because it includes variable time-delays in the range of 250–400 ms. The dynamics between the movement of bucket and pile is even more complex and as a result, no high-fidelity models exists for closed-loop control.

The hydraulic pressure in the cylinders (which provide lift/tilt forces used as input features) are affected both by the reaction forces on the bucket by the pile and the actions of the operator, executed $\sim 200–400$ ms ago. Thus, in order to produce lift/tilt actuator commands in real-time, the model needs to do time-ahead prediction. We use a TDNN architecture to accomplish this, which incorporates the dynamics present in the system to produce an appropriate actuator command for the lift and tilt joysticks. The TDNN model implements a moving window of delayed inputs to capture the dynamics.

In this experiment, the total delay length (TDL) and delay step (DS) are selected from \{160, 320, 640\} ms and \{20, 40, 80, 160\} ms, respectively, to produce six models of two different size with a constant number of parameters ($n_P$). The description of these models with their given name is presented in Table 4.

The results of delay variation ($N = 6$) in the input layer are presented in Fig. 12. In this experiment, we use the classification model as shown in Fig. 7b with 45% throttle in the third phase. It is observed that increasing the TDL value (including more history) in the model decreases the bucket-filling time. The results show that the model uses long term (>320 ms) patterns in the data to predict better control signals. This appears related to the
Figure 12: Effect of delay variation in the input layer of the classification network. Refer to Table 4 for further information about the axis labels. A model with longer history of machine data performs significantly better than a model based on a shorter history. This implies that the delay step (DS) does not play any important role. Models with more parameters where TDL = \{160, 320\} ms perform better, however this is not true for TDL = 640 ms.
presence of a variable delay of 250–400 ms in the wheel-loader hydraulics system. For smaller value of TDL, the model with more parameters (L160, L320) gives better performance. This suggests that high flexibility provides an advantage in this regime where the TDL is smaller than the delay in hydraulics. But when TDL = 640 ms, a model with more parameters (L640) does not provide an advantage over a smaller model (S640).

With k-fold \((k = 12)\) cross-validation, it is found that the six classification models and the regression model perform about equally well in the simulations, in terms of the root-mean-square-error (RMSE, Eq. 6) between the operator \((y_i)\) and the predicted control actions \((\hat{y}_i)\). However, the field test performance of the six models differ considerably. A plot of 12-fold cross-validation error for the six models for lift/tilt predictions is shown in Fig. A1 in the Appendix.

\[
RMSE = \sqrt{\frac{\sum_{i=1}^{N} (\hat{y}_i - y_i)^2}{N}} \quad (6)
\]

We conclude that minimizing the RMSE in search for a better model for this problem is not recommended. Instead, the model should be optimized on the performance metrics defined by the field experts.

4.5. Model vs expert operator

The training data was logged in a controlled environment (pile located under a roof) with specific instructions to the operator to maintain a constant engine RPM during the bucket-filling process. The last experiment carried out focuses on comparing the performance of one model with the expert operator. During this experiment, the operator is asked to use the machine like in a production scenario. The analysis of the control actions reveal that the operator is more aggressive with the throttle in this case, reaching up to 70% of full throttle at the end of bucket filling. As a result, the operator managed to finish most bucket-fillings in less than 7 s.

Fig. 13 shows the result of a comparison test between the chosen model (L640) and the operator. We conclude that the neural-network model is similar to the expert operator in terms of bucket weight, with slightly longer bucket-filling time. The longer bucket-filling time is likely because the neural-network model is operating at constant throttle of 55%, while the operator is modulating the throttle to avoid lift and translation stalls.
4.6. Control actions

The control actions produced by the operator and the model are time-series signals of different lengths. Dynamic Time Warping (DTW) is a method used to compare and find patterns in time-series of different lengths [43]. The DTW distance between two time-series is a measure of how similar the two time-series are. The DTW distances of control actions produced by the operator reveals how similar the actions of the operator are between different bucket-fillings, and similarly for the control actions produced by the model between different trials of automatic bucket-filling.

Fig. 14 illustrates the DTW distances for lift and tilt control actions produced by the operator and the model in the comparison test performed in section 4.5. The scooping that is the most similar to all other scoopings, determined by the minimal sum of DTW distances to all other scoopings, is chosen as the reference for both the operator and the model. It can be seen that the control signals produced by the model are more similar to each other (smaller DTW distances) compared to the operator control actions. In Fig. 15, the reference bucket-filling example used for calculating the DTW distances in Fig. 14, is illustrated for both the model and the expert operator. The model output and operator actions are not particularly similar, but the
The average value of lift and tilt DTW distance for the model shows that the control actions produced by the model are more uniform across trials compared to the operator. However, the variance in the lift and tilt DTW distances for the model shows that the model is not repeating the same control actions during each bucket-filling.

Model still manages to fill the bucket efficiently. This supports the expert know-how in the field, which suggests that there is not a single way to fill the bucket. The slightly longer bucket-filling time of the model is likely related to the lower magnitude of the actions produced by the algorithm, especially the throttle, towards the end of the bucket-filling process.

### 4.7 Failures

The bucket-filling algorithm based on the classification model presented in this paper has been successful in all the 136 trials carried out. But before that, there were many unsuccessful trials, which are also interesting to analyze.

The first unsuccessful attempt to make a complete bucket-filling algorithm was based on only three scooping phases (phase two was omitted). The neural network was started immediately after phase one, when the pile is detected. The reason for the failure of this approach is that the first lifting action by the operator occurs even before phase two starts, and thus it was never included in the training data. This occurs because an experienced operator anticipates the delay in the hydraulics and acts pro-actively with control signals. One idea for how to solve this problem is to train and start the
model earlier than the first lift action, and by having a longer TDL (1–2 s) in
the TDNN network. There are a few risks associated with this approach; (1)
the possibility that the network will start lifting before the pile is reached and
that a failure mode is built into the system, and (2) if the network doesn’t
produce a high lift action, wheel-spin may occur which damages the tires.

The neural-network trained on data recorded in dry and controlled medium
gravel pile fails to perform bucket-filling in (1) wet and compact material,
and (2) pile with a long and low slope. It suggests that different networks
are needed to be trained with data collected in different conditions. Then,
with a collection of networks trained in different conditions, a suitable model
can be used for the present condition.

5. Conclusions and future work

Automation of construction, mining and quarry industry require automatic
bucket-filling functions for front-end loaders. Modeling the pile and
the bucket-pile interactions is considered an intractable problem and thus
traditional closed-loop control is not possible. Operators use their vision,
sound and vestibular system to perform the bucket filling process efficiently.
In this paper an imitation learning model trained on expert operator data
is presented, analyzed and demonstrated with field trials. We find that a
time-delayed neural network (TDNN) architecture with input data obtained
from the machine is sufficient to efficiently perform the bucket-filling task au-
tomatically. The neural-network based bucket-filling algorithm for a wheel-
loader loading medium coarse gravel performs slightly worse than our expert
operator with correct bucket weights and only 26% longer bucket filling time.
We find that a classification-based model performs better than a classical
regression model because the operators tend to modulate the joysticks be-
 tween different levels. The total delay time (TDL) parameter of the TDNN
model is found to be an important hyperparameter for this task due to the
variable delay present in the hydraulics of the wheel-loader. An TDL value
higher than the value of the delay in the hydraulics results in smaller bucket-
filling time. The analysis of the middle layer weights show that all of the
input features used to train the model are significant for the prediction of
the control actions.

The work presented here can be improved in several different directions.
The bucket-filling algorithm can be built to handle different type of pile
shapes and materials by training different models for different environments
and selecting the best model based on sensing (for example by point-cloud
scans), prior to starting the algorithm. It is likely that the model trained
on one type of material (such as the medium coarse gravel considered here)
should not work as good as an expert operator on another pile like soil. With
a reinforcement learning algorithm, such as the deep-deterministic policy gra-
dient, the neural-network parameters can be tuned after each bucket-filling
to maximize a reward function based on the chosen performance metrics. In
this way, the neural-network can adapt to changing conditions and over time
and improve its performance.

Acknowledgment

This work has been has been done with support from the program "For-
donsstrategisk Forskning och Innovation, FFI", Vinnova grant 2017-01958.
Fredrik Sandin’s contribution is supported by the Kempe Foundations under
contract Gunnar Öquist Fellow 2014. We thank Thomas Lind who is the
expert operator in this study.
References


Appendix

<table>
<thead>
<tr>
<th></th>
<th>Weight (tons)</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train 1</td>
<td>7.62±0.33</td>
<td>12.83±0.59</td>
</tr>
<tr>
<td>Train 2</td>
<td>7.45±0.24</td>
<td>11.44±0.49</td>
</tr>
<tr>
<td>Train 3</td>
<td>7.31±0.13</td>
<td>10.37±0.16</td>
</tr>
</tbody>
</table>

Table A1: Performance of three different training runs of one model type with $N = 6$. The models in this experiment are based on the classification architecture with $TDL = 160$ ms, $DS = 20$ ms, such that $n_P \sim 800$. In this experiment 45% throttle is used.
<table>
<thead>
<tr>
<th>Throttle</th>
<th>Weight (tons)</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>45%</td>
<td>6.84±0.19</td>
<td>7.66±0.27</td>
</tr>
<tr>
<td>50%</td>
<td>6.81±0.10</td>
<td>6.98±0.26</td>
</tr>
<tr>
<td>55%</td>
<td>7.20±0.08</td>
<td>7.78±0.10</td>
</tr>
</tbody>
</table>

Table A2: Performance of two different model type with variation in throttle with \( N = 6 \) levels. The models in this experiment are based on the classification architecture with \( TDL = 640 \) ms, \( DS = 80 \) ms, such that \( n_P \sim 500 \).

Figure A1: Root-mean-square-error (RMSE) and their average values and spread in quartiles obtained by cross-validation (12-fold) procedure for the six classification and one regression model on the operator data. The outliers are shown with the plus (‘+’) symbol.