The 18th Scandinavian International Conference on Fluid Power, SICFP'23, May 30- June 1, 2023, Tampere, Finland

Towards Multiple Material Loading for Wheel Loaders Using Transfer Learning

Daniel Eriksson , Reza Ghabcheloo , and Marcus Geimer[*]

Faculty of Engineering and Natural Sciences, Tampere University, P.O. Box 1001, 33014, Finland
E-mail: daniel.eriksson@tuni.fi, reza.ghabcheloo@tuni.fi
[*]Institute Mobile Machines, Karlsruhe Institute of Technology, Rintheimer Querallee, 76131, Germany
[*]E-mail: marcus.geimer@kit.edu

Abstract

Automatic bucket filling is an important puzzle piece in reducing the impact of labor shortage in the construction industry by lowering the learning curve for new operators and reducing the required skill-level. This task has proven to be challenging to automate, because it is difficult to model all the interaction forces between the wheel loader and the loaded material. This has led to machine learning approaches receiving a lot of attention in recent years. However, previous works have only focused on loading one specific material, while, in this paper we investigate the problem of loading multiple materials. Our controller is a neural network with a 1D-CNN architecture. The controllers are trained using recorded data from expert operators loading three different materials, namely, sand, gravel and blasted rock, and evaluated on a real machine. We compare three different cases. In the first case, three controllers were synthesized using training data from only one material type. In the second case, two controllers were synthesized by combining training data from all materials. One controller used 100% of the training data and the other used a third of the available training data. Finally, in the third case, four controllers per material were synthesized with transfer learning by retraining the last layer of the controllers trained in case 1 and 2. Nine controllers in total were evaluated on each material type by measuring the loading time and the weight in the bucket after each scoop. Our experiments show promising results with transfer learning as an approach for loading several different bulk materials and efficiently use a limited amount of training data. The best result was achieved by using blasted rock as a base material and then transfer learning to adopt the neural network to the other materials. Furthermore, the best controllers achieved human comparable performance for each material. On the other hand, combining materials into a single controller proved to not yield satisfying results. They had poor or mediocre performance on all materials.

Keywords: Bucket Filling, Wheel Loader, Neural Networks, Imitation Learning, Learning from Demonstrations, Transfer Learning

1 Introduction

Adding assistance functions to help the operator to automate tasks for heavy duty mobile machines (HDMM) is in general becoming more important when trying to improve fuel and production efficiency. These improvements can cut operating costs as well as solve the emerging problem of labor shortage in the construction and mining industries by helping less experienced operators achieve high productivity [1, 2].

Wheel loaders are popular HDMMs because of their versatility and capability to perform various tasks with different attachments. The most typical task for a wheel loader is to load and unload different materials, where the materials can range from everything between very fine, loose, and homogeneous materials, like sand, to very rough and non-homogeneous, materials like blasted rock. The materials have very different properties such as kernel size, shape, roundness, density, etc, from each other and therefore, they require slightly different loading techniques in order to achieve optimal performance in terms of fill-grade, loading time and fuel consumption. An

Figure 1: An L576 wheel loader loading sand [3].

experienced wheel loader operator can load different materials fast and efficient with their intuition. However, it is not straight forward to develop a controller with conventional rule based control techniques because of unknown material parameters and unknown dynamics in the interaction forces between the wheel loader and the pile. Previous work using classical control theory and trajectory optimization have not resulted in clear success [4, 5]. Machine learning approaches have therefore gained traction and popularity in recent years as a solution to the bucket filling problem [6–9]. However, the focus have mostly been on loading a single material, and when using the same controller to load a different material the performance was degraded [10], which is to be expected since different materials require slightly different bucket filling techniques and machine learning approaches have a hard time generalizing to inputs outside of the distribution used for training. This assumption breaks for the bucket filling task when the training data comes from one material but the controller is loading a different one. However, there are a lot of similarities between loading two different materials and therefore it would be advantageous to use and share knowledge between them. This is referred to as transfer learning (TL) in the literature where one approach is to retrain the last layer of a neural network by providing additional samples from the new material [11]. This can also be achieved by using reinforcement learning (RL) and let the wheel loader self-optimize on a new material as in [8].

Transfer learning is advantageous in applications where the amount of labeled training data is limited or costly to acquire, as in our case with real machines, because TL is able to adapt an existing neural network to a new material with minimum number of new training samples. This approach is necessary in our use-case because wheel loaders encounters very different materials during normal operation and the effort to collect large amounts of training data for each material does not scale and is unfeasible.

This paper introduces a transfer learning approach to multiple material loading evaluated on blasted rock, gravel and sand. Figure 2 shows the different strategies used in this paper and how they use the training data. The most straightforward approach is to collect the necessary amount of training data from all of the target materials and train a specific controller for each of them. This comes with the caveat of having n controllers for n materials and the need to collect the same amount of training data from all the materials. A more efficient use of the training data is achieved with transfer learning where the controller trained on one material adopts to a new material with only a small amount of new training data by retraining the last layer of the neural network controller. This also results in n controllers for n materials, but with significantly lower amounts of required training data. Another approach is to use the same training data but create one controller capable of loading all the target materials by combining the training data and train a single neural network. However, different materials needs different techniques but the neural network outputs the average loading technique from the training data, and therefore might not output the optimal trajectory for any material.

A related approach to transfer learning is multitask learning (MTL), which has been successful in natural language processing, face recognition [12–14]. In MTL several distinct tasks are combined in the same network but with a unique output neuron for each task. The idea is that the network will have access to more data and therefore learn

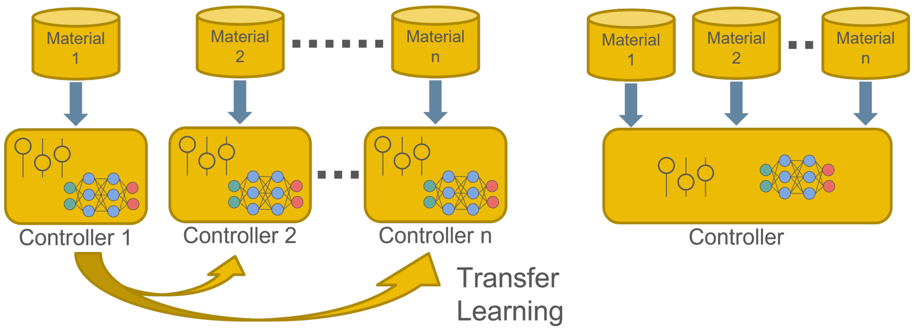


Figure 2: Overview of different strategies for loading multiple materials. The left-side option is to collect data and train one controller for each material. The necessary data recording effort might be reduced with transfer learning. The right-side option is to combine all the materials into a single dataset and controller.

better representations of the inputs and achieve higher performance and robustness. This would result in a network with n outputs for n materials. It is also possible to add one extra output as an auxiliary task during training, for example in our case, the type of material being loaded. The extra output is only relevant during training for biasing the shared hidden layers to increase the predictive performance. Therefore, the extra output does not fulfill any purpose when the network is deployed as a controller [15]. However, MTL and the previously mentioned RL approach have not been explored and are out of scope for this paper.

In the next section, the neural network controller and its architecture is explained. The materials and training data used in this study is described in the section after that. Lastly the results are presented and a conclusion is discussed in the final sections.

2 Neural Network Controller

Imitation learning (IL) or Learning from demonstrations (LfD) is a method to learn an end-to-end controller for robotics systems by leveraging demonstrations recorded by humans experts. This paper uses the popular behavior cloning framework as an imitation learning approach to train a neural network controller [16]. The inputs to the network consists of five signals from the machine: the angle of the bucket θ_{tilt} and boom θ_{lift}, force in the tilt and lift cylinders F_{tilt}, F_{lift}, and the machine velocity v. The neural network then maps these five measurements to three outputs commands: u_{tilt}, u_{lift}, $u_{throttle}$ that are sent to wheel loader. The inputs and outputs of the neural network controller can be seen in fig. 3 where the outputs from the neural network is a direct replacement of the operator joystick and throttle pedal signals. Furthermore, each input channel consists of 16 samples of the signal, sampled with a time window of 225 ms at 15 ms rate, and also normalized to fit the range of -1 to 1.

The controller consists of a combination of a 1 dimensional convolutional neural network (1D-CNN) and a fully connected output network. All the layers are activated with the rectified linear unit (ReLU) activation function except for the last layer which is activated by the hyperbolic tangent (tanh) function in order to map the output to -1, 1 for full control of boom and bucket. The full architecture of the network is illustrated in fig. 3.

CNN uses the convolution operation between the input and a kernel to transform the data and find patterns. The CNN is most noticeably used in image processing, recognition and object detection with a 2D kernel operating on image data [17]. It has also been used with 1D kernel in the past for time series data [18–20].

The networks used in this paper was developed with the PyTorch framework [21] and trained to predict the operator commands in the training data using the aforementioned inputs. The networks were optimized with the ADAM optimizer [22] and with respect to the mean square error (MSE) between the predicted and true operator command.

A relatively shallow architecture was chosen because of the limited amount of training data available and the CNN was chosen to be able to use the temporal information in the signals and at the same time reducing the number of parameters needed to be trained. Furthermore, the CNN approach showed good performance in earlier experiments motivating the use of the same architecture for this experiment as well. Lastly, this architecture has in total 2641 trainable parameters.

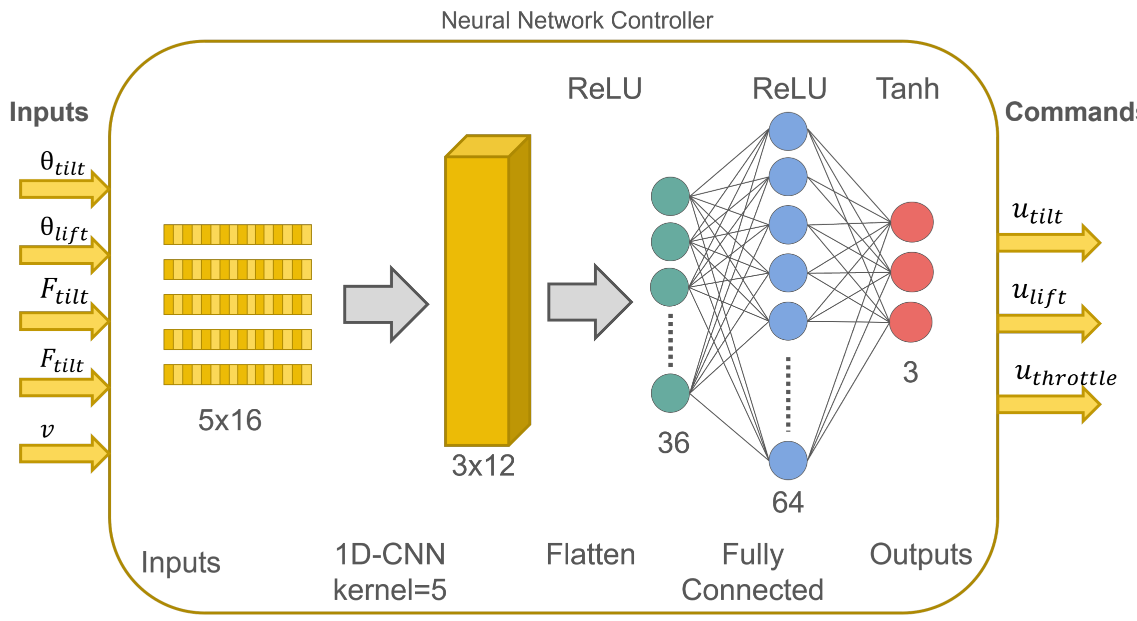


Figure 3: 5 signals are sent to the neural network controller as inputs. The bucket and boom angles: θ_{tilt}, θ_{lift}, forces in tilt and lift cylinders: F_{tilt}, F_{lift} and the machine velocity v. The 3 outputs, u_{tilt}, u_{lift}, $u_{throttle}$, of the neural network controls the tilt and lift joystick commands as well as the throttle pedal, respectively. The convolutional layer uses a kernel size of 5 with stride of 1 and 3 filters. This output is flattened and sent to a fully connected layer with a hidden dimension of 64. All the layers use the ReLU activation function except for the last one which uses tanh.

3 Method

A Liebherr L576 wheel loader was used for this paper for recording data and testing the synthesized controllers. Figure 1 shows an example of a general image of a Liebherr L576 loading sand. Pressure sensors and sensors for measuring the length of the tilt and lift cylinders, which are required for calculating the relevant forces and angles, are already available on the machine. Thus, no additional sensors or equipment except for a standard consumer grade PC with a CAN-bus interface was used for this paper. The PC was used for running the bucket filling algorithm described in section 3.2, in a ROS node, as well as receiving sensor data from the machine.

3.1 Training data

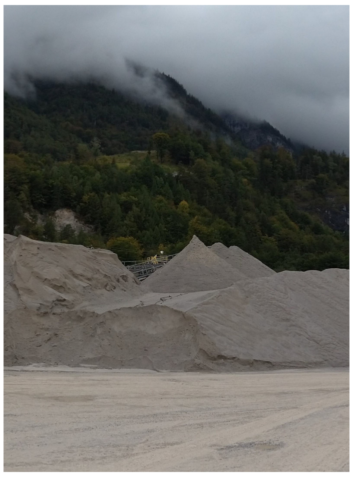
We recorded in total 306 bucket fillings spread equally over 3 piles with different materials. The following materials were selected: sand (0-4 mm), gravel (0-64 mm), and blasted rock (0-200 mm). Figure 4 shows the three different material piles and a close up of the materials with a shoe as reference for the kernel size. These materials were selected for this experiment because they are commonly used in quarries and construction sites, and they also have distinct features and therefore require slightly different loading strategies for an efficient and full bucket. The three selected materials represents an increasing complexity and non-homogeneous characteristics, where sand is the least complex and the most homogeneous, and blasted rock is the most complex and non-homogeneous because of the varied sizes of particles.

The training data was recorded during a period of one month at the same quarry by an expert operator during normal operation. Figure 5 shows the mean and standard deviation of the operators weight and loading time for bucket fillings collected from *blasted rock*, *gravel* and *sand*. *combined_33*, and *combined_100* represent a combination of all the three materials. *combined_33* contains 34 samples from each pile selected at random with 102 samples in total. *combined_100*, on the other hand, contains all the recorded bucket fillings with 306 samples in total.

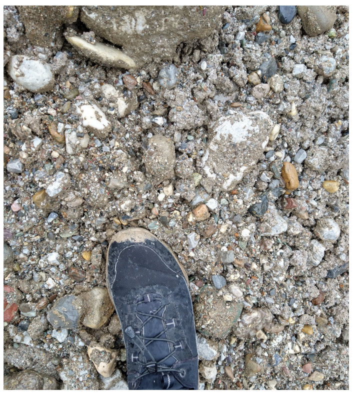
The training data shows that the operator is able to fill the bucket with similar loading times and material weights for different materials, but with relatively high standard deviation. The slight difference in loaded mean weight of different materials can be explained by the differences in the densities between them. Furthermore, the loading time is also very similar between materials, usually 5-8 seconds to fill the bucket, where the loading time is defined as the time the operator spends digging. The loading time starts when the force in the lift cylinders are higher than a threshold, signifying that the bucket has penetrated the pile, and ends when tilt and lift cylinders are extended to a certain degree or when the operator switches gear to reverse, signifying that the bucket has left the pile.

(a) *Blasted rock pile.*

(b) *Gravel pile.*


(c) *Sand pile.*

(d) *Close up of the blasted rock pile.*


(e) *Close up of the gravel pile*

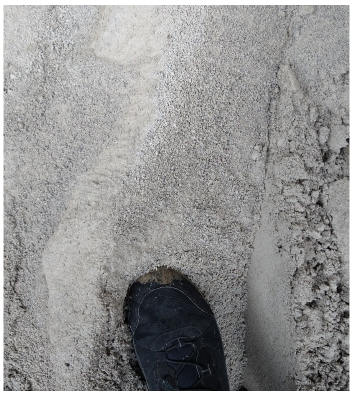

(f) *Close up of the sand pile*

Figure 4: The three different material piles with an overview image and a close up image.

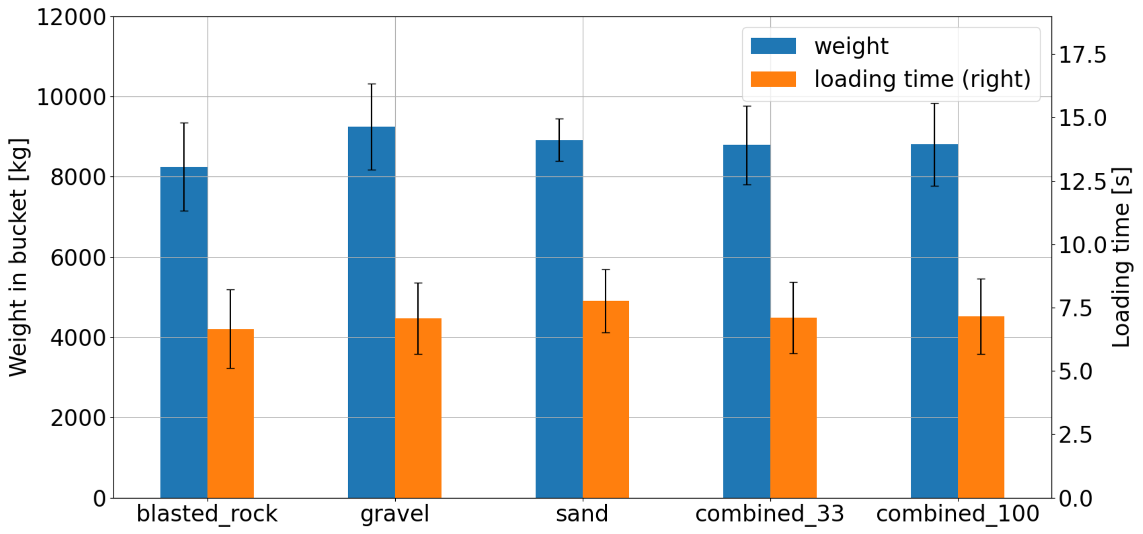


Figure 5: Weight and time distributions of the five different collected datasets. The height of each bar corresponds to the mean value and the error bar to the standard deviation of the dataset. The combined_33 and combined_100 contains a combination of the three other datasets with an inclusion ratio of 33% and 100% respectively.

3.2 Bucket filling algorithm

We divide the bucket filling into 3 phases as illustrated in fig. 6, similarly to the approach in [6, 8]. With this approach the neural network controller is only in control of the most challenging stage in the bucket filling process, that is, phase 2. It also has the advantage of simplifying the output commands from the neural network because the network does not need to know when it is in the pile by itself, it assumes that it is in pile when being activated. Furthermore, this resolves the problem where the controller might start tilting and lifting the bucket before it has reached the pile. Lastly, the bucket filling algorithm assumes that the wheel loader is a few meters in front of the pile and that no steering is required to reach it.

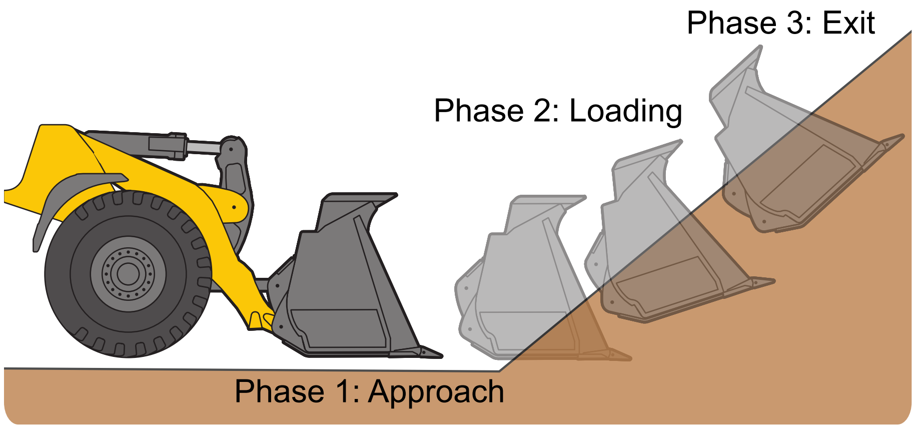


Figure 6: The three different digging stages. The illustration of the wheel loader is inspired from [23].

1. *Approach*: The goal of the first phase is to bring the wheel loader into loading position in the pile. The bucket is automatically lowered and placed flat on the ground and the wheel loader is accelerated with a constant throttle value of 50% towards the pile. When the bucket penetrates the pile the force in the lift cylinders increase until it reaches a threshold and the algorithm switches to the next phase.

2. *Loading*: During this phase the θ_{tilt}, θ_{lift}, F_{tilt}, F_{lift}, and v signals are sent to neural network controller which determine the commands u_{tilt}, u_{lift}, and $u_{throttle}$ from the outputs of the neural network controller which are sent to the wheel loader via the CAN-bus. The output of the neural network is in the range of -1 to 1, meaning it is capable of both lifting and lowering the boom, and tilting the bucket inwards and outwards. However, the throttle command is restricted to forward motion only, by clipping the output. The next phase starts when the tilt or lift cylinder lengths are extended to a certain degree.

3. *Exit*: This phase makes sure that the bucket is fully tilted inwards so that no material gets spilled from the bucket. The bucket is also automatically raised in order for the built-in weighing system to weigh the mass of the material in the bucket.

3.3 Controllers

We synthesized 17 controllers in total from the 3 datasets described in section 3.1, the selection of the controllers was chosen in accordance with fig. 2. The controllers are grouped into 3 categories depending on which training data was used for the training: specialized, combined, and transfer. The categories are explained in the next sections. A summary of all the controllers and the training data used in the synthesizing is described in tab. 1.

3.3.1 Specialized controllers

Three specialized controllers, namely *blst_rock*, *gravel* and *sand*, were trained with only one material type. Training data for each controller included 102 samples. These controllers should perform the best when evaluated on the same pile as the training data.

3.3.2 Combined controllers

Two controllers, namely *cmb100* and *cmb33*, were synthesized by training on the *combined_100* and *combined_33* datasets, respectively. These controllers will evaluate the generalization capabilities of the neural network by testing if it is possible to distill the knowledge from several bucket filling techniques or if the output will be an average of the techniques and thus have poor performance on all materials.

Table 1: Overview of the synthesized controllers and the training data used. The specialized controllers were trained on one material each. The combined controllers used a mixture of the materials as training data. The transfer controllers used the previous controllers as a basis and transfer learning to adopt to a different material. The transfer controllers use the naming convention: <base_controller>_tl_<target_material>.

		No. bucket fillings			
Type	Controller	Total	Blasted rock	Gravel	Sand
Specialized	blst_rock	102	102		
	gravel	102		102	
	sand	102			102
Combined	cmb100	306	102	102	102
	cmb33	102	34	34	34
Transfer	blst_rock_tl_gravel	112	102	10	
	blst_rock_tl_sand	112	102		10
	gravel_tl_blst_rock	112	10	102	
	gravel_tl_sand	112		102	10
	sand_tl_blst_rock	112	10		102
	sand_tl_gravel	112		10	102
	cmb100_tl_blst_rock	316	112	102	102
	cmb100_tl_gravel	316	102	112	102
	cmb100_tl_sand	316	102	102	112
	cmb33_tl_blst_rock	112	44	34	34
	cmb33_tl_gravel	112	34	44	34
	cmb33_tl_sand	112	34	34	44

3.3.3 Transfer controllers

Additionally, 12 more controllers were synthesized using the specialized and combined controllers as a basis for transfer learning. The last layer of the previous controllers were retrained with only 10 samples from the target material. The synthesized transfer learning controllers use the naming convention: <*base_controller*>_tl_<*target_material*>, for example, *blst_rock_tl_gravel* uses the *blst_rock* as a base controller and transferred to the *gravel* material by using 10 samples for retraining.

3.4 Evaluation metrics

We want to find a high performing and robust controller by measuring and evaluating three metrics: weight in bucket, loading time, and success rate. The weight of the material in the bucket is used as a proxy for fill-grade because it is easier to measure the weight in the bucket than the volume with the available sensors on the machine. A downside with this approach is that the material density influences the weight of a full bucket for different materials. Therefore, the weight is always compared to the human operator for each material pile. The loading time metric is defined as the time the wheel loader is in phase 2 where the neural network is in control. This definition also ignores different starting distances from the pile and only measures the interesting part of the bucket filling process. Lastly, a successful bucket fill is defined as the controller reaching phase 3 and therefore completing the loading.

All of these metrics are important, and a controller needs to be able to achieve a minimum performance and robustness criterion in order for it to be considered as a viable method. It has to achieve a full bucket in under 10.0 seconds with a 100% success rate. A weight of 800 kg is considered as a full bucket with the the specific bucket and target materials used in the tests. Finally, the mean weight and loading time should preferably be comparable to human level performance. In other words, the controller should be able to always achieve a full bucket load in a timely manner.

4 Results

This section presents the closed-loop performance of the developed controllers on the same material piles as the collected training data. For each material a total of 9 controllers were evaluated: all of the specialized and combined controllers and the relevant transferred controllers. For the transferred controllers, only the controllers transferred to the specific material being evaluated were tested. For example, when evaluating the controllers on the blasted rock material only the transfer controller that did not use *blst_rock* as a base controller but blasted rock as a target material were evaluated.

The mean weight and loading time μ_{weight}, μ_{time} as well as the standard deviation σ_{weight}, σ_{time} are calculated over a sample size of 5 trials for each of the controllers, and are reported in Table 2 and in fig. 7. Furthermore, fig. 7 shows the controllers sorted from left to right with the highest loaded material weight on the left. The controllers left of the red dotted line achieved the aforementioned minimum performance and robustness criterion.

Table 2: Reported mean and standard deviation for the material weight and the loading time for each controller evaluated on the three piles. No evaluation of controller on a pile is denoted with a -. NaN denotes the complete failure of a controller on a pile.

Material	**Blasted Rock**				Gravel				Sand			
Controller Score	μ_w	σ_w	μ_t	σ_t	μ_w	σ_w	μ_t	σ_t	μ_w	σ_w	μ_t	σ_t
blst_rock	9005	667	8.45	0.73	9562	429	8.99	0.47	9757	646	7.51	0.87
gravel	5665	1882	9.49	4.13	8521	671	6.02	0.47	8213	718	5.15	0.47
sand	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
cmb100	7710	526	6.62	0.85	8515	500	9.63	2.28	7758	233	6.34	0.89
cmb33	7812	1191	12.39	3.27	8125	2651	11.91	5.57	NaN	NaN	NaN	NaN
blst_rock_tl_gravel	-	-	-	-	10521	671	6.99	0.52	-	-	-	-
blst_rock_tl_sand	-	-	-	-	-	-	-	-	9834	908	6.94	0.33
gravel_tl_blst_rock	7913	823	7.14	0.99	-	-	-	-	-	-	-	-
gravel_tl_sand	-	-	-	-	-	-	-	-	5893	316	5.04	0.36
sand_tl_blst_rock	8481	495	9.44	0.56	-	-	-	-	-	-	-	-
sand_tl_gravel	-	-	-	-	5743	1556	4.55	0.87	-	-	-	-
cmb100_tl_blst_rock	8171	234	7.19	0.97	-	-	-	-	-	-	-	-
cmb100_tl_gravel	-	-	-	-	8984	178	6.97	1.94	-	-	-	-
cmb100_tl_sand	-	-	-	-	-	-	-	-	8075	885	9.99	2.39
cmb33_tl_blst_rock	6271	1958	5.19	0.36	-	-	-	-	-	-	-	-
cmb33_tl_gravel	-	-	-	-	NaN	NaN	NaN	NaN	-	-	-	-
cmb33_tl_sand	-	-	-	-	-	-	-	-	NaN	NaN	NaN	NaN
Units	[kg]	[kg]	[s]	[s]	[kg]	[kg]	[s]	[s]	[kg]	[kg]	[s]	[s]

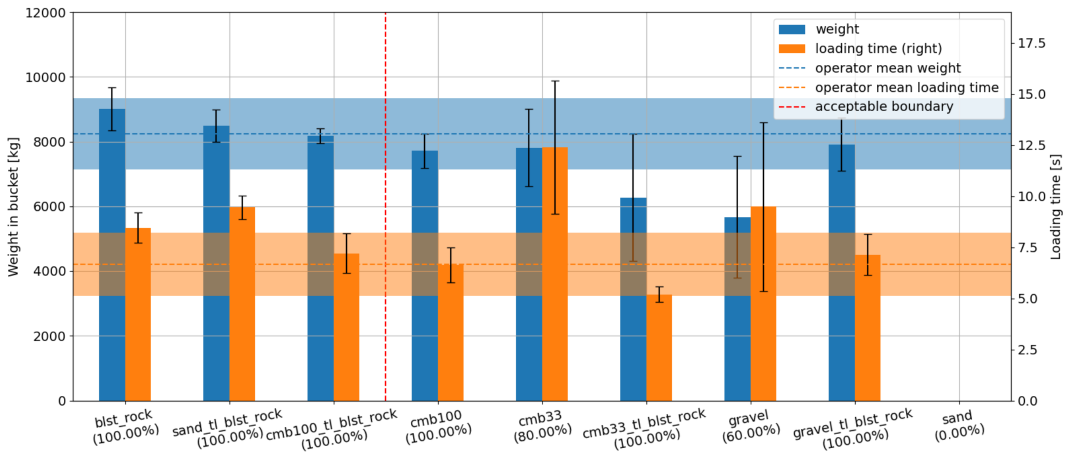
As mentioned in section 3.4, a successfully bucket fill is defined as the controller reaching phase 3. The common failure modes for not reaching this phase are the following: wheel loader started to lift itself up with the boom and bucket, excessive wheel spin and the wheel loader getting stuck in the pile. When any of these failure mode occurred the experiment was manually aborted and the trial was counted as a fail.

4.1 Specialized controllers

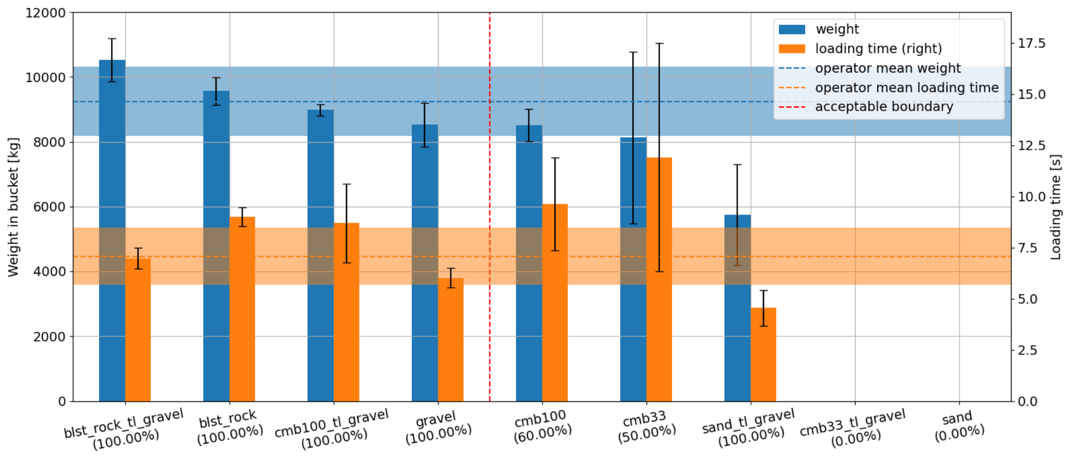
Figure 7 shows that the specialized controllers except for the *sand* controller had a high and robust performance on their specific material. The *sand* controller did not succeed in filling a bucket from any pile because the wheel loader either got stuck in pile or using too much throttle commands resulting in excessive wheel spin. The *gravel* controller was the fastest on both the sand and gravel piles but with less weight than the operator mean and some other controllers. The *blst_rock* controller was the only one of the specialized controllers to reach the minimum requirements on the blasted rock pile. It was also the best performing controller overall on that pile. Furthermore, it also had a good performance on the gravel pile.

4.2 Combined controllers

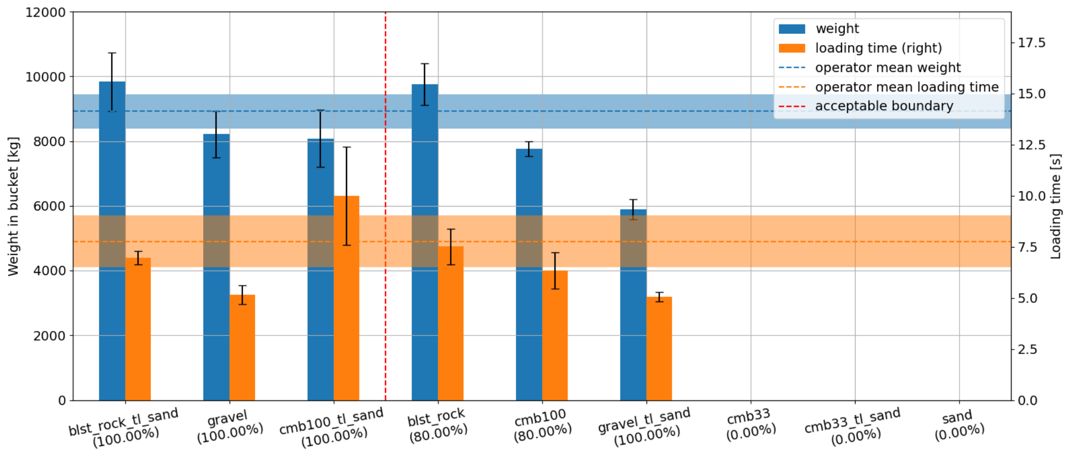
The combined controllers did not have a good performance or robustness in general since none of the *cmb100* or *cmb33* reached the minimum performance and robustness requirement on any of the piles as is evident from fig. 7. The best of the combined controllers was *cmb100* on blasted rock. It succeeded every time to fill the bucket quickly but sometimes with lower than 8000 kg of material. Furthermore, the *cmb100* performed better and more robust than the *cmb33* on all materials. The failure mode for these controllers were excessive wheel spin, getting stuck in the pile or lifting itself up with the boom.



(a) Weight and time distributions of the controllers evaluated on the blasted rock pile



(b) Weight and time distributions of the controllers evaluated on the gravel pile



(c) Weight and time distributions of the controllers evaluated on the sand pile

Figure 7: Weight and time distributions of the different controllers evaluated on each of the three target materials. The x-axis contains the name and success rate of the controller. The controllers left of the red dotted line have acceptable performance and robustness according to the definition of: 100% success rate, higher weight than 8000 kg, faster than 10 seconds. The acceptable controllers are sorted by weight with the leftmost having the highest weight.

4.3 Transfer controllers

The transfer controller could improve the performance and robustness in most cases. In fig. 7b the *blasted_rock* controller was improved with transfer learning to both increase the material weight and reduce the loading time to human comparable levels. It it shown that the *blst_rock* base controller transferred to the target material could achieve highest weight in a timely manner for all the investigated materials. The figure also show that the *cmb100* was improved with transfer learning to reach the minimum performance and robustness criterion. Lastly, fig. 7a shows that the transfer learning approach could improve the *sand* controller to successfully load the blasted rock.

5 Conclusions and Outlook

This paper explores a transfer learning approach for loading of three distinct materials with a 1D-CNN neural network controller using training data recorded by an expert operator. The results shows that the best use of training data for synthesizing controllers with high performance and robustness is to first train on a non-homogeneous material like blasted rock and then use transfer learning to learn loading of a specific material. This strategy minimizes the data collection effort since it only needs 10 additional training samples from a new material in order to achieve high robustness and performance on it. Furthermore, it is also possible to first combine all the training data and then use transfer learning to adopt the controller to the target material, but this requires more data to be collected and lower performance than the first option.

The combined controllers did not generalize to all of the different bucket filling techniques, but averaged the actions and had mediocre performance on all of the materials. When combining training data from multiple materials it might be necessary to use a larger network in order catch the different dynamics between the material and bucket. Having one controller being capable of loading all materials can be an advantage in some applications if the material to be loaded is unknown. It would also be interesting to use multitask learning where the neural network has different heads for each material, i.e the number of outputs from the network would be 3 times n materials but only the 3 relevant outputs would be used for each material. These strategies were not tested in this paper but they might work better than our approach, and could be interesting to explore in the future.

The specialized controller performed well on the same materials as the training data, except for the *sand* controller, which failed in loading all of the materials. This can be explained by the homogeneity and low complexity of sand, and therefore the controller is only exposed to a limited state space and does not learn to generalize well to any material. This can also be explained by overfitting on the training data caused by similar reasons. Furthermore, it proves that it is better to use training data from more complex materials because of a higher degree of explored state space.

The promising results of the transfer learning approach shows that it's also interesting to explore reinforcement learning for loading different and unknown materials. This approach is similar to the transfer learning approach used in this paper, but instead of retraining the last layer of the controller with additionally recorded bucket fillings from an operator, it is retrained by the wheel loader exploring automatically the best strategy on its own.

Acknowledgment

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No. 858101.

References

[1] Bobbie Frank, Lennarth Skogh, Reno Filla, Anders Froberg, and Mats Alaküla. On increasing fuel efficiency by operator assistance systems in a wheel loader.

[2] S. Dadhich, U. Bodin, and U. Andersson. Key challenges in automation of earth-moving machines. *Automation in Construction*, 68:212–222, 2016.

[3] Liebherr. L 576 xpower. `https://www.liebherr.com/de/aut/produkte/baumaschinen/erdbewegung/radlader/details/241926.html`. Accessed: 2022-10-14.

[4] R. Filla and B. Frank. Towards finding the optimal bucket filling strategy through simulation. In *Proceedings of 15:th Scandinavian International Conference on Fluid Power, 15th Scandinavian International Conference on Fluid Power, Fluid Power in the Digital Age, SICFP'17, June 7-9 2017 - Linköping, Sweden*, Linköping Electronic Conference Proceedings, pages 402–417. Linköping University Electronic Press, 2017.

[5] Heshan A. Fernando, Joshua A. Marshall, Håkan Almqvist, and Johan Larsson. Towards controlling bucket fill factor in robotic excavation by learning admittance control setpoints. In Marco Hutter and Roland

Siegwart, editors, *Field and Service Robotics*, volume 5 of *Springer Proceedings in Advanced Robotics*, pages 35–48. Springer International Publishing, Cham, 2018.

[6] Siddharth Dadhich, Ulf Bodin, Fredrik Sandin, and Ulf Andersson. Machine learning approach to automatic bucket loading. In *2016 24th Mediterranean Conference on Control and Automation (MED)*, pages 1260–1265. IEEE, 21/06/2016 - 24/06/2016.

[7] E. Halbach, J. Kämäräinen, and R. Ghabcheloo. Neural network pile loading controller trained by demonstration. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 980–986, 2019.

[8] Siddharth Dadhich, Fredrik Sandin, Ulf Bodin, Ulf Andersson, and Torbjörn Martinsson. Field test of neural-network based automatic bucket-filling algorithm for wheel-loaders. *Automation in Construction*, 97:1–12, 2019.

[9] Nataliya Strokina, Wenyan Yang, Joni Pajarinen, Nikolay Serbenyuk, Joni Kämäräinen, and Reza Ghabcheloo. Visual rewards from observation for sequential tasks: Autonomous pile loading. *Frontiers in Robotics and AI*, 9, 2022.

[10] Siddharth Dadhich, Fredrik Sandin, Ulf Bodin, Ulf Andersson, and Torbjorn Martinsson. Adaptation of a wheel loader automatic bucket filling neural network using reinforcement learning. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9. IEEE, 19/07/2020 - 24/07/2020.

[11] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[12] Michael Crawshaw. Multi-task learning with deep neural networks: A survey.

[13] Joachim Bingel and Anders Søgaard. Identifying beneficial task relations for multi-task learning in deep neural networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 164–169, Valencia, Spain, 2017. Association for Computational Linguistics.

[14] Zhanpeng Zhang, Ping Luo, Chen Change Loy, and Xiaoou Tang. Facial landmark detection by deep multi-task learning. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, volume 8694 of *Lecture Notes in Computer Science*, pages 94–108. Springer International Publishing, Cham, 2014.

[15] Rich Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997.

[16] Takayuki Osa, Joni Pajarinen, Gerhard Neumann, J. Andrew Bagnell, Pieter Abbeel, and Jan Peters. An algorithmic perspective on imitation learning. *Foundations and Trends in Robotics*, 7(1-2):1–179, 2018.

[17] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)*, pages 1–6. IEEE, 2017.

[18] Kevin J. Lang, Alex H. Waibel, and Geoffrey E. Hinton. A time-delay neural network architecture for isolated word recognition. *Neural Networks*, 3(1):23–43, 1990.

[19] Sajjad Abdoli, Patrick Cardinal, and Alessandro Lameiras Koerich. End-to-end environmental sound classification using a 1d convolutional neural network. *Expert Systems with Applications*, 136:252–263, 2019.

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

[21] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[22] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization.

[23] Technical Documentation Department LBH. *Wheel Loader Operator's Manual*. Liebherr-Werk Bischofshofen GmbH, Bischofshofen, Austria, 10 2021.

Acronyms

CNN	convolutional neural network
HDMM	heavy duty mobile machine
IL	imitation learning
ML	machine learning
MSE	mean square error
MTL	multitask learning
RL	reinforcement learning
TL	transfer learning

Nomenclature

Designation	Denotation	Unit
θ_{tilt}	Bucket angle	rad
θ_{lift}	Boom angle	rad
F_{tilt}	Force in tilt cylinder	N
F_{lift}	Force in lift cylinder	N
v	Velocity of wheel loader	m/s
μ_{weight}	mean of weight	kg
μ_{time}	mean of loading time	s
σ_{weight}	standard deviation of weight	kg
σ_{time}	standard deviation of loading time	s