2024 European Control Conference (ECC)
June 25-28, 2024. Stockholm, Sweden

Impact of Data Quantity and Composition on Bucket Filling Performance for Wheel Loaders

Daniel Eriksson[1], Reza Ghabcheloo[1], and Marcus Geimer[2]

Abstract— **This paper investigates the impact of training data with different quantities and compositions on the performance and robustness of a Neural Network (NN) controller for the wheel loader bucket filling task. Collecting training data for machine learning methods with a real-world Heavy Duty Mobile Machine (HDMM) is expensive, and therefore knowing how to collect the data and in what quantities will significantly reduce the data collection effort. We collected 2000 bucket fillings of non-homogeneous material, more specifically, a blasted rock pile with a kernel size of 0–400 mm. No previous study has reported such a challenging material composition. The collected data was divided into 6 datasets with sizes of 10, 20, 50, 100, 500, and 2000 bucket fillings. We use the Dynamic Time Warp (DTW) distance, k-medoids clustering, and the silhouette score, to create diverse and dissimilar datasets. Furthermore, one additional dataset was created with 10 bucket fillings, which are as similar as possible, resulting in 7 datasets in total. The datasets were used to synthesize 7 controllers that were then evaluated with a set of experiments to compare their performance to one another and the human operator. The results showed that the controller trained on similar bucket fillings was not robust and had poor performance, as expected. The experiment also showed that all the controllers trained on diverse data were robust enough to load the blasted rock material. However, the loaded material weight was less than the human operator, where the best controller loaded 9% less material weight, but 11% faster than the human operator.**

I. INTRODUCTION

Delivering autonomous functions to different kinds of Heavy Duty Mobile Machines (HDMMs) is an ongoing research area for both universities and companies alike. The wheel loader is a typical HDMM that is used in quarries and construction sites, where a common task is to move material between different locations. During this task, filling the bucket requires the most experience from the operator for efficient completion. A low-skilled operator will be slower, use more fuel, and load less material in the bucket compared to an experienced operator [1]. Therefore, a bucket filling assistance system could make operators more fuel-efficient and productive. It would also reduce the necessary skill level of operators, which is important with the current trend of labor shortages in the construction industry [2], [3].

Creating a rule-based algorithm or a trajectory-following controller is challenging since the interaction forces between the material and the bucket are hard to model with many unknown parameters that cannot be efficiently determined in the field. Therefore, machine learning (ML) methods are currently the most popular approaches for the bucket filling problem [2], [4]. Imitation learning has been used to create Neural Network (NN) controllers for the bucket filling task in previous research [4]–[9]. These works have shown that it is possible to load materials such as sand, different kinds of gravel, and blasted rock (0–200 mm) with human-level performance using around 100 examples [5], [8]–[10]. However, this paper targets blasted rock (0–400 mm), which is a non-homogeneous material and more challenging to load compared to the previous materials.

The training data is the most important part of any ML system, and a model is only as good as the data it is trained on. It is also a well-established fact that more data yields better performance and generalization of ML models, as well as reducing the problem of overfitting. It is easy to collect more training data in domains such as natural language and image recognition tasks, where there are many open datasets as well as the internet [11]. This is a more challenging problem when HDMMs are involved, since it is necessary to record data on real machines, which is very time-consuming and expensive. Therefore, it is important to know how much training data is necessary to collect and what composition the training data should have.

The composition of the training data is also important; it should be diverse to create NN models with high generalization capabilities and to avoid bias in them. For example, a cat classifier should include many examples of different cats for the classifier to have high accuracy on unknown data. For example, increasing the diversity with augmented data has been demonstrated to improve the generalization capabilities for image classification [12], [13]. The same conclusions should apply to bucket filling data, where the dataset should consist of different but optimal bucket filling examples. This results in a more exhaustive exploration of the input and output spaces and minimizes the risk of a covariate shift between the recorded data and the inputs at inference time [14]. The covariate shift can occur even if we evaluate the controller on exactly the same machine and material pile as the training data because the pile could be slightly different at test time, and compound errors from the controller's outputs causes the input distribution to drift from the original data. Diverse training data will also reduce the risk of overfitting since the variance of the inputs is higher, thus increasing the generalization capabilities [15].

The goal of this paper is to investigate how the quantity

*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.

[1]Automation and Mechanical Engineering, Faculty of Engineering and Natural Sciences, Tampere University, P.O. Box 1001, 33014, Finland daniel.eriksson@tuni.fi

[2]Institute Mobile Machines, Karlsruhe Institute of Technology, Rintheimer Querallee, 76131, Germany

Fig. 1: The blasted rock (0-400 mm) pile used for data collection and evaluation.

and composition of training data used for synthesizing bucket filling controllers affect their performance and robustness. Thereby, find the best strategy for recording bucket filling data in the future. This is tested by splitting the training data into datasets with different sizes and compositions, and then using them to train different NN controllers. The synthesized controllers are compared to each other and the human operator in a field test.

II. TRAINING DATA

We collected training data from a blasted rock material (0-400 mm), shown in Fig. 1, of approximately 2000 bucket fillings. The training data were collected over the course of 2 months with several expert operators during normal working operations.

The blasted rock material used for these experiments has a larger kernel size of 0-400 mm in comparison to 0-200 mm, used in our previous work [9]. This material also had a higher ratio of larger to smaller boulders compared with the previous blasted rock material. These properties make the material challenging, and it takes an experienced wheel loader operator to load a full bucket.

The data consists of sensor data and operator command signals, and it was sampled at 15 ms intervals. The sensor data $\boldsymbol{x} = (\theta_{tilt}, \theta_{lift}, F_{tilt}, F_{lift}, v)$, where θ_{tilt} and θ_{lift} are the tilt and lift angles, F_{tilt} and F_{lift} are the force in the tilt and lift cylinders, and v is the velocity of the machine, as visualized in Fig. 2. The joint angles θ_{tilt}, and θ_{lift}, are 0 when the bucket is placed flat on the ground, and increases in the direction of the arrows in Fig. 2.

The operator commands $\boldsymbol{u} = (u_{tilt}, u_{lift}, u_{throttle})$ were also recorded from the joystick and pedal signals. The joystick signals u_{tilt} and u_{lift}, controls the wheel loader's bucket by actuating the tilt and lift hydraulic cylinders via the machine's internal controller. The joystick commands are in the range of $[-1, 1]$. Lastly, $u_{throttle}$ controls the engine's power output, where the signal is in the range of $[0, 1]$.

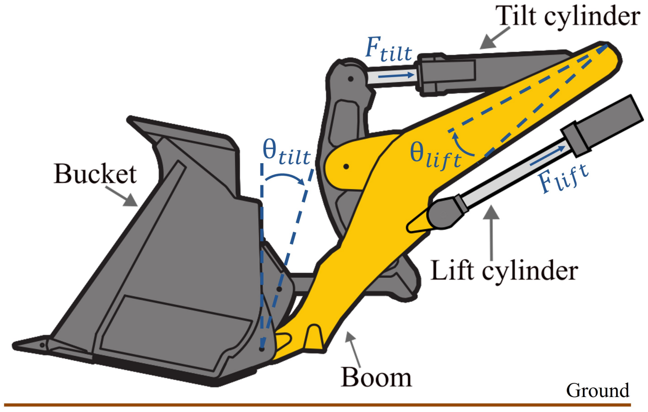


Fig. 2: Definition of the joint angles and forces.

III. BACKGROUND

This section describes the background knowledge and methods used for measuring the similarities between the bucket fillings and for dividing the recorded dataset into different sizes. The collected bucket fillings forms a set of M time series $\boldsymbol{T} = \{\boldsymbol{\tau^1}, \boldsymbol{\tau^2}, ..., \boldsymbol{\tau^M}\}$, where $\boldsymbol{\tau^i} = (\tau^i_1, \tau^i_2, ..., \tau^i_I)$ is a time series with length I.

A. Dynamic Time Warp

We use Dynamic Time Warp (DTW) [16] to measure the similarities between different bucket filling time series. DTW was introduced for speech applications as an optimal match problem, and is a suitable distance measure for time series of different lengths, as in our case, or for equal but shifted time series. Another common distance measurement is the Euclidean distance, but it does not have the properties described above that are necessary for our data [16], [17].

The DTW distance between two time series, $\boldsymbol{\tau^i} = (\tau^i_1, \tau^i_2, ..., \tau^i_I)$ of length $|\boldsymbol{\tau^i}| = I$, and $\boldsymbol{\tau^j} = (\tau^j_1, \tau^j_2, ..., \tau^j_J)$ of length $|\boldsymbol{\tau^j}| = J$, is defined as an optimization problem on the form:

$$\begin{aligned} \mathrm{DTW}(\boldsymbol{\tau^i}, \boldsymbol{\tau^j}) \quad &= \min_{\boldsymbol{\pi}} \sqrt{\sum_{k=1}^{K} d(\tau^i_{\pi_{k,i}}, \tau^j_{\pi_{k,j}})^2} \\ \text{s.t.} \quad & \pi_1 = (1,1), \\ & \pi_K = (I, J), \\ & \pi_{k,i} \leq \pi_{k+1,i} \leq \pi_{k,i} + 1, \\ & \pi_{k,j} \leq \pi_{k+1,j} \leq \pi_{k,j} + 1, \end{aligned} \tag{1}$$

and finds the minimal warping path $\boldsymbol{\pi} = (\pi_1, \pi_2, ..., \pi_K)$, of length $|\pi| = K$, where $\pi_k = (\pi_{k,i}, \pi_{k,j})$. Here $\pi_{k,i}$, and $\pi_{k,j}$ are indices from the two time series, with $\pi_{k,i} \in \mathbb{N}$, $1 \leq \pi_{k,i} \leq I$, and $\pi_{k,j} \in \mathbb{N}$, $1 \leq \pi_{k,j} \leq J$. The function $d(\boldsymbol{\tau^i}, \boldsymbol{\tau^j})$ is any valid distance function, and we use the Euclidean distance in this paper. The first and last points of the time series must be matched, which is guaranteed by the two first constraints. The warping path must also be monotonically increasing and continuous, which is guaranteed by the last two constraints.

B. k-medoids

k-medoids is a clustering algorithm that clusters a set of objects into k discrete clusters. The set of objects can be any abstract, which in the context of this paper, is the set $\boldsymbol{T}$ that contains the bucket filling time series as defined earlier. The algorithm is similar to the more known k-means clustering algorithm, but with the difference that k-medoids uses actual objects as centers, called medoids. It also finds efficient solutions using any arbitrary dissimilarity measure, for example, the DTW distance defined in the previous section. k-means clustering, on the other hand, usually requires the Euclidean distance.

The k-medoids algorithm greedily selects k time series as medoids. The rest of them are associated with the closest medoid and the association is swapped iteratively to lower the dissimilarity between them in the clusters. The result is a set of k clusters $\boldsymbol{C} = \{\boldsymbol{C_1}, \boldsymbol{C_2}, ..., \boldsymbol{C_K}\}$, where each cluster $\boldsymbol{C_K} \subset \boldsymbol{T}$, and $\boldsymbol{C_K} = \{\boldsymbol{\tau^{K_1}}, \boldsymbol{\tau^{K_1}}, ..., \boldsymbol{\tau^{K_n}}\}$, contains an unspecified amount of n time series, where the number of objects are different for each cluster [18], [19].

C. Silhouette Score

The silhouette score is used to evaluate how well each object belongs to its own cluster compared to the other clusters. A high silhouette score means that the object is very similar to the other objects in its own cluster, while a low silhouette score means that the object is more similar to objects in a neighboring cluster. It is calculated using the average dissimilarities between the object and the objects in its own cluster, in relation to the average dissimilarities between the object and the objects in the closest neighboring cluster [20].

The silhouette score for an object $\boldsymbol{\tau^i} \in \boldsymbol{C_A}$, where cluster $\boldsymbol{C_A} \in \boldsymbol{C}$ with size $|\boldsymbol{C_A}|$, is defined as the ratio:

$$s(\boldsymbol{\tau^i}) = \frac{a(\boldsymbol{\tau^i}) - b(\boldsymbol{\tau^i})}{\max\{a(\boldsymbol{\tau^i}), b(\boldsymbol{\tau^i})\}}, \quad (2)$$

$$a(\boldsymbol{\tau^i}) = \frac{1}{|\boldsymbol{C_A}| - 1} \sum_{\boldsymbol{\tau^j} \in \boldsymbol{C_A}, \boldsymbol{\tau^i} \neq \boldsymbol{\tau^j}} \delta(\boldsymbol{\tau^i}, \boldsymbol{\tau^j}), \quad (3)$$

$$b(\boldsymbol{\tau^i}) = \min_{B \neq A} \frac{1}{|\boldsymbol{C_B}|} \sum_{\boldsymbol{\tau^j} \in \boldsymbol{C_B}} \delta(\boldsymbol{\tau^i}, \boldsymbol{\tau^j}), \quad (4)$$

where $\boldsymbol{C_B} \in \boldsymbol{C}$ is a neighbor cluster of size $|\boldsymbol{C_B}|$, and is defined as the cluster different from $\boldsymbol{C_A}$ which has the lowest average dissimilarity between the objects in $\boldsymbol{C_B}$ and the object $\boldsymbol{\tau^i}$. Furthermore, the function $\delta(\boldsymbol{\tau^i}, \boldsymbol{\tau^j})$, is any valid dissimilarity function defined between object $\boldsymbol{\tau^i}$ and another object $\boldsymbol{\tau^j}$, which is in our case, the DTW distance defined in (1).

In summary, the silhouette score is calculated by first calculating the mean dissimilarities between $\boldsymbol{\tau^i}$ and all other objects in $\boldsymbol{C_A}$ in (3). Then the average dissimilarities between the object $\boldsymbol{\tau^i}$, and the objects $\boldsymbol{\tau^j}$ neighbor cluster $\boldsymbol{C_B}$, are calculated by (4).

IV. Related Work

Choosing samples from a dataset has been researched before in the ML context, mostly as a form of reducing the dataset size to speed up the training of ML algorithms. Therefore, most research concerns how to select a subset of the training data that has the same properties as the original data, and where an ML algorithm trained on the subset will have the same results on the test set [21]–[24]. Even though the core problem statement is different from ours, we share some common goals, like selecting diverse and dissimilar examples.

Coresets are a group of methods for summarizing a large dataset, resulting in a subset with the same characteristics as the original dataset. A coreset is usually defined for the target objective function that minimizes it on the coreset instead of the original. Therefore, the coresets found in the literature are often specialized to the target problem and are often not general solutions [21], [22].

Subspace clustering is a related approach, and it is the process of selecting appropriate clusters of similar objects in a dataset with a defined similarity function. Subspace clustering algorithms usually use clustering methods such as k-means or k-medoids, in combination with a similarity measurement, similar to our approach [25]. One of these algorithms is Dissimilarity-Based Sparse Subset Selection (DS3) [23]. DS3 finds a representative subset of the original data using pairwise dissimilarities between the data points in the source and target sets. The authors demonstrated that it could find a better subset of the data compared to k-medoids and a few other clustering methods [23].

A similar but different problem is dataset distillation, which reduces a dataset by compressing it into a new dataset with a smaller size. The new dataset is not a subset of the original data but an encoded version of it, and thus not interesting for us to consider [24].

The methods described above could be adapted to fit with our problem statement, but we chose a simpler and more straightforward method as a first step, which is described in the next section.

V. Method

The first step is to record many bucket filling examples and subsample them into smaller datasets. There are several approaches for subsampling a dataset, as discussed earlier. The naive approach is to select randomly or chronologically, but in this case, it will not be possible to conclude if any performance differences are due to the size or composition of the dataset. As mentioned earlier, training data with high diversity is known to improve the performance of NN models [12], [13]. Therefore, we want to create a dataset where the input data has a high level of dissimilarity between the bucket filling examples. The overall dissimilarity should be the same for each dataset, where the major difference is the dataset's size. This would make it possible to isolate and evaluate the effect of the dataset's size on performance.

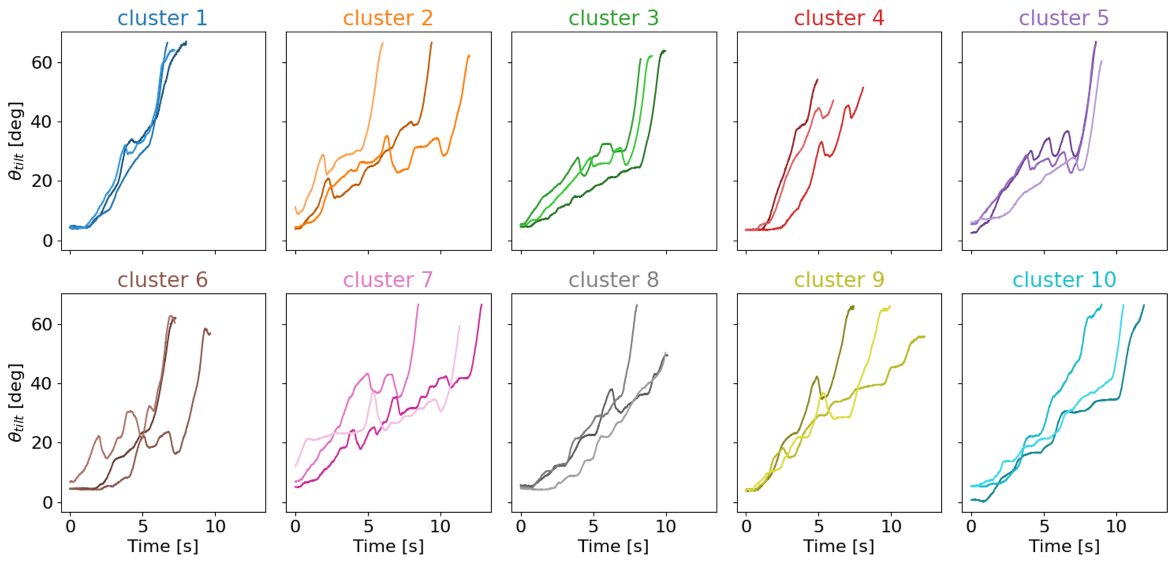


Fig. 3: Three bucket fillings examples from each cluster using θ_{tilt} for visualization.

A. *Creating datasets*

The selection process for diverse and dissimilar subsampled datasets with sizes, $m_1, ..., m_x, ..., m_X$, is summarized by the following steps:

1) **Measure dissimilarities:** Calculate the DTW distance between all the bucket filling examples and create a symmetric dissimilarity matrix.
2) **Cluster examples:** Use the dissimilarity matrix to cluster the bucket filling examples into k clusters with the k-medoids method. k is chosen as the size of the smallest dataset, m_1.
3) **Calculate silhouette scores:** Calculate the silhouette score for each bucket filling example and sort them from highest to lowest score within each cluster.
4) **Select samples:** Draw n samples with the highest silhouette score from each of the k clusters to create a dataset of size $m_x = nk$. Repeat this step to create datasets of different sizes by changing n.

We also use a similar method to create a dataset with as similar bucket fillings as possible by drawing n samples from one of the clusters created in step 2.

VI. Experimental Setup

Here we describe the data collection, the created datasets and their respective synthesized controllers, as well as the wheel loader setup used for the experiments.

A. *Datasets*

The 2000 recorded bucket filling were used to create 6 datasets of different sizes: $m_1 = 10$, $m_2 = 20$, $m_3 = 50$, $m_4 = 100$, $m_5 = 500$, and $m_6 = 2000$, using the method explained in section V. The details for each step are explained below:

1) Measure dissimilarities: We used the variables: θ_{tilt}, θ_{lift}, and v for calculating the DTW distance between all the bucket filling examples. These variables were chosen because they represent the trajectory of the bucket tip through the pile. Furthermore, the python implementation from [26] was used to calculate the DTW distances.

2) Cluster examples: We used the FasterPAM algorithm introduced in [27] for k-medoids clustering with a value of $k = 10$ because our smallest dataset, m_1, consists of 10 bucket fillings. An example of three bucket fillings from each cluster is shown in Fig. 3 using θ_{tilt} for visualization. The figure shows that the bucket fillings in each cluster look more similar to each other than to the other clusters. However, there are also some similarities between the clusters because we do not want to find the true underlying cluster structure. We only want to find the 10 most dissimilar bucket fillings, since our smallest dataset has the size $m_1 = 10$.

3) Calculate silhouette scores: The silhouette score is visualized for all the bucket filling examples in Fig. 4. The figure shows the size of each cluster, the average silhouette score for each cluster, and the average silhouette score for all examples. The silhouette score indicates a weak clustering structure, meaning that some clusters could be combined. However, this is not a problem in our use case because of the reasons mentioned in the previous paragraph.

4) Select samples: We created 6 datasets with different numbers of bucket fillings by drawing the n bucket fillings with the highest silhouette score from each cluster. For example, to create the dataset with size $m_2 = 20$, we select the $n = 2$ bucket fillings with the highest silhouette scores from each cluster, and so on for the other datasets. We also created a 7th dataset with as similar bucket fillings as possible by selecting 10 bucket fillings with the highest silhouette scores from cluster $\boldsymbol{C_{10}}$. We chose $\boldsymbol{C_{10}}$ because it had the highest average silhouette score, thus containing the most similar bucket fillings.

B. *Controllers*

We used the same NN controller architecture as in our previous work [8], [9], which is shown in Fig. 6.

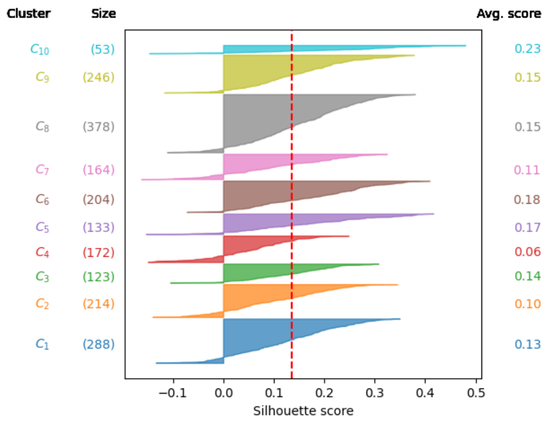


Fig. 4: Silhouette score for all the bucket filling examples in each cluster. The red line indicates the average silhouette score for all examples, which was 0.14.

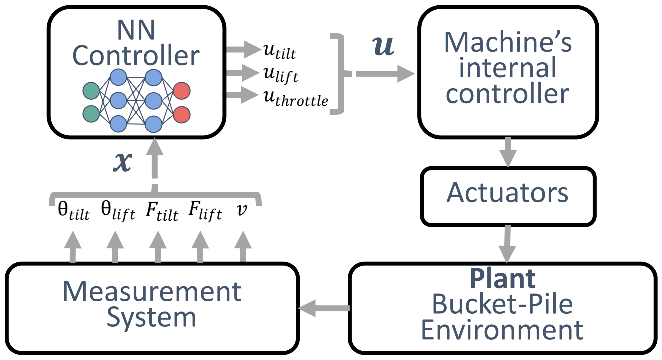


Fig. 5: Block diagram of the control structure on the machine. The NN controller commands $\boldsymbol{u}$, controls the tilt and lift hydraulic valves as well as the engine power via the machine's internal controller.

The NN controller takes 16 samples of $\boldsymbol{x}$, as defined in Section II, as input, corresponding to the last 225 ms. The input tensor is thus of the shape 16×5 and the NN controller outputs $\boldsymbol{u}$, also defined in Section II. An overview of the control structure is shown in Fig. 5, where the commands $\boldsymbol{u}$ controls the hydraulic valves of the tilt and lift cylinders as well as the engine power via the machine's internal controller.

The first layer of the NN model is a 1-D Convolutional Neural Network (1D-CNN) with a kernel size of 5. The output of this layer is flattened and relayed to a fully connected layer with a hidden size of 64 neurons. The final layer is activated by the hyperbolic tangent function (tanh) so the outputs are in the range [-1,1]. All the other layers are activated by the Rectified Linear Unit (ReLU) function.

The network is trained to predict the true operator commands in the training data using supervised learning, and the mean squared error is used as a loss function with the ADAM optimizer [28]. We also add a weight decay factor, λ, to the loss function in order to minimize the risk of overfitting, using a value of $\lambda = 0.0001$.

The 7 datasets from the previous section were used to

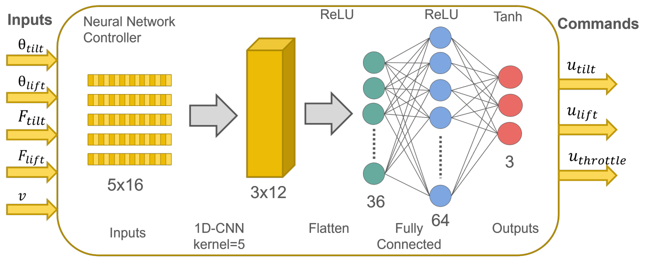


Fig. 6: Architecture of the NN controller. The model receives the joint angles, forces, and velocity, and outputs the predicted commands.

synthesize 7 controllers trained on each dataset with the architecture above. The NN models and the training procedure were implemented in PyTorch [29].

The NN controllers: *d-10, d-20, d-50, d-100, d-500* and *d-2000* were trained on the 6 datasets with the corresponding size in the name using the format *d-<size>*. The last controller *s-10*, was trained on the dataset with 10 bucket fillings from the same cluster.

C. Bucket Filling Algorithm

The bucket filling loading algorithm used in this paper is the same algorithm as in our previous work [8], [9], which divides the task into three phases: approach pile, loading, and exit. The first phase makes sure that the bucket is located inside the pile before digging by laying it flat on the ground and moving forward with 50% throttle. The second phase starts when F_{lift} reaches a threshold. Then the NN controller is activated and fills the bucket until a θ_{tilt} threshold is reached. Finally, the last phase makes sure that the bucket is fully tilted in and that the material in the bucket is weighed.

D. Wheel loader setup

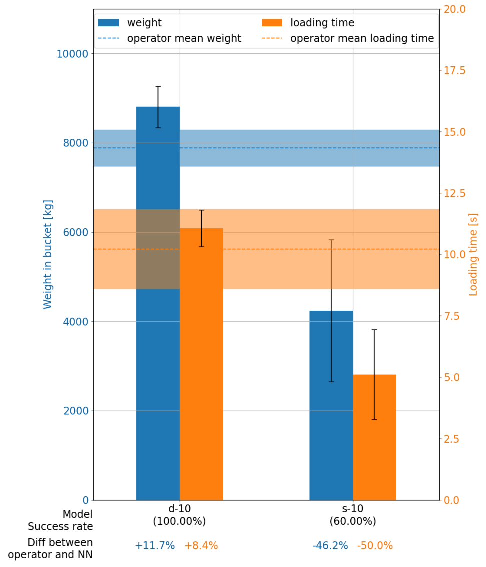
We used a 33-tonnes Liebherr L586 wheel loader for data collection as well as evaluating the models. The machine was controlled with a consumer laptop connected to the wheel loader's control unit via the CAN-bus.

VII. EXPERIMENTS

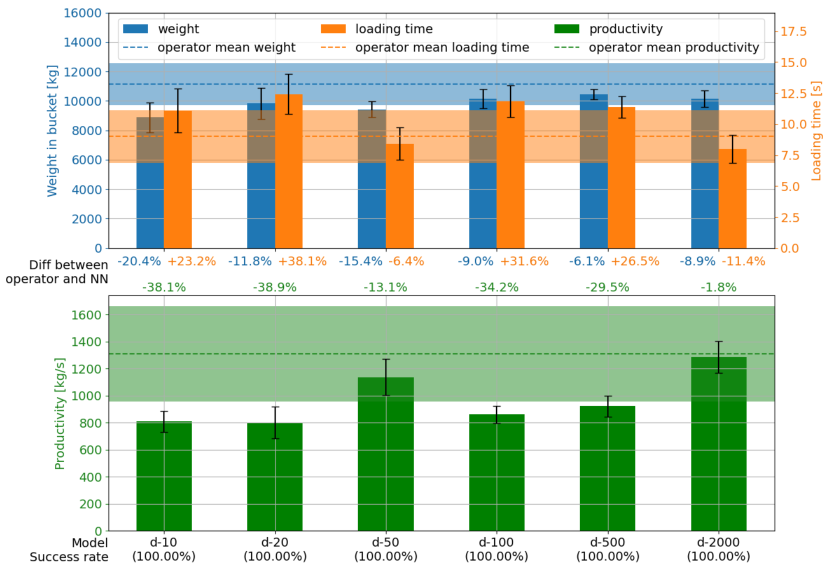
The controllers were tested and evaluated using two experiments. Firstly, we validated our subsampling method and compared the controller *d-10*, which was trained on a diverse dataset, with *s-10*, which was trained on a dataset with similar bucket fillings. The second experiment evaluates the performance of the controllers trained on different data quantities.

Figure 7 shows the results from the two experiments, where we tested the relevant controllers 10 times for each experiment and reported the mean and standard deviation. Furthermore, the material is always dumped in the same location, to ensure that the pile is as similar in shape and structure as possible between the different trials.

We also evaluate the robustness of the controllers; a controller capable of loading the bucket successfully every time is defined as robust. A bucket filling is considered



(a) Weight and time distributions of the *d-10* and *s-10* controllers evaluated on a gravel pile.



(b) Weight and time distributions of the controllers evaluated on the blasted rock pile. The green bars and lines represent the productivity [kg/s] during the bucket filling.

Fig. 7: Performance and robustness of the controllers evaluated in the two experiments. The x-axis displays the controller, the success rate, and the difference between the average operator performance and the average controller performance. Each vertical bar is the average of 10 bucket fillings per controller, and the horizontal lines show the average operator performance. Blue represents the average weight [kg] and orange represents the average loading time [s].

successful if the controller is able to load the bucket without getting stuck in the pile. The wheel loader might get stuck in the pile if the tilt or lift cylinders are stalled, which depends on the output commands from the controller. Another failure mode is excessive wheel spin caused by too high throttle commands.

A. *Data composition*

We first tested how the composition of the training data affects the robustness and performance of the synthesized controller by evaluating the *d-10*, and *s-10* controllers on a gravel pile. This pile was different from the blasted rock pile used to collect the training data because it was more available for testing. Using a gravel pile for this experiment is not a problem because we showed in [9] that gravel is a simpler material to load, and using a more complex material for training is advantageous.

Figure 7a shows the average performance and robustness of 10 trials for the two controllers evaluated on the gravel pile, as well as the human operator's performance. The figure shows that the *d-10* controller, which was trained on a dataset composed of diverse bucket fillings, has a 100% success rate. It is also able to get more material weight than the operator, but with a slightly longer loading time. The *s-10* controller, which was trained on similar bucket fillings, only had a success rate of 60%, and a performance far below that of the human operator.

B. *Quantity of data*

For the second experiment, we tested how the quantity of the data influences the digging performance of the controllers. We evaluated the 6 controllers: *d-10, d-20, d-50, d-100, d-500* and *d-2000*, on the blasted rock pile that was used for data collection, and the results are shown in Fig. 7b. The figure shows that all the controllers are robust with 100% success rate, even the controller that was trained using 10 bucket filling examples. The results also show that the performance is increasing with more training data, which we can see from the mean productivity bar graphs in green. Furthermore, the results show that using 100 examples or more does not significantly improve the loaded weight; it stays around 9% lower than the human operator, and using more examples only improves the loading time.

None of the controllers are able to reach the same digging performance as the human operator. The best-performing controller, *d-2000*, is able to load on average around 10 tonnes in 7.9 s, compared to the human operator with about 11 tonnes in 9 s. In other words, it gets 8.9% less material but is 11.4% faster than the human operator and has a productivity that is 1.8% lower. Productivity in this context is a measurement of how much material weight is moved per time unit [kg/s].

VIII. CONCLUSIONS

We demonstrate how diverse and dissimilar bucket filling examples can improve the robustness and stability of a NN controller, and how the performance is improving with an increasing number of examples. We create diverse datasets of different sizes by measuring the similarities between the bucket fillings and choosing the ones with high dissimilarities.

The results showed that a diverse dataset of different bucket fillings is more important than its size. A diverse dataset is also crucial for robust performance, as shown in the first experiment. Furthermore, we achieved a robust controller, *d-10*, with 100% success rate using only 10 bucket filling examples. This controller was also able to load gravel with human-level performance. The *s-10* controller, which was trained on similar bucket fillings, had, on the other hand, a 60% success rate and low overall performance. Therefore, when collecting bucket fillings for imitation learning, it is important to include as many dissimilar but optimal bucket fillings as possible. One possibility is to record data from piles with different shapes as well as using different expert operators since different humans load the bucket in a slightly different way.

The results show that the performance is increasing with a larger dataset, which is expected, but using a dataset larger than 100 bucket fillings mostly improves the loading time and not the material weight. Furthermore, none of the controllers could achieve the same level of loaded material as the human operator, even using 2000 examples. However, the material used - blasted rock (0-400 mm) - is a complex material and very challenging to load, and thus our approach to imitation learning may have reached its limits, and more training data might not help. This is indicated by the marginal performance increase using the largest dataset for training.

Finally, the controller was robust, and the loaded material weight was about 9% lower than the human operator when using 100 bucket fillings or more as training data. This makes it possible to use other methods to further fine-tune the network and improve its performance on the pile. One method is to use Reinforcement Learning (RL), which has been used in [10], to optimize and improve an existing NN controller on a new material.

ACKNOWLEDGMENT

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

REFERENCES

[1] B. Frank, L. Skogh, R. Filla, A. Froberg, and M. 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," *Automat. in Construction*, vol. 68, pp. 212–222, 2016.

[3] B. Brucker Juricic, M. Galic, and S. Marenjak, "Review of the construction labour demand and shortages in the eu," *Buildings*, vol. 11, no. 1, p. 17, 2021.

[4] S. Dadhich, U. Bodin, F. Sandin, and U. Andersson, "Machine learning approach to automatic bucket loading," in *2016 24th Mediterranean Conf. on Control and Automat. (MED)*. IEEE, 2016, pp. 1260–1265.

[5] S. Dadhich, F. Sandin, U. Bodin, U. Andersson, and T. Martinsson, "Field test of neural-network based automatic bucket-filling algorithm for wheel-loaders," *Automat. in Construction*, vol. 97, pp. 1–12, 2019.

[6] E. Halbach, J. Kämäräinen, and R. Ghabcheloo, "Neural network pile loading controller trained by demonstration," in *2019 Int. Conf. on Robot. and Automat. (ICRA)*, 2019, pp. 980–986.

[7] W. Yang, N. Strokina, N. Serbenyuk, J. Pajarinen, R. Ghabcheloo, J. Vihonen, M. M. Aref, and J.-K. Kamarainen, "Neural network controller for autonomous pile loading revised," in *2021 IEEE Int. Conf. on Robotics and Automat. (ICRA)*. IEEE, 2021, pp. 2198–2204.

[8] D. Eriksson and R. Ghabcheloo, "Comparison of machine learning methods for automatic bucket filling: An imitation learning approach," *Automat. in Construction*, vol. 150, p. 104843, 2023.

[9] D. Eriksson, R. Ghabcheloo, and M. Geimer, "Towards multiple material loading for wheel loaders using transfer learning," in *Proc. of 18th Scand. Int. Conf. on Fluid Power, SICFP23*, 2023.

[10] S. Dadhich, F. Sandin, U. Bodin, U. Andersson, and T. Martinsson, "Adaptation of a wheel loader automatic bucket filling neural network using reinforcement learning," in *2020 Int. Joint Conf. on Neural Networks (IJCNN)*. IEEE, 2020, pp. 1–9.

[11] A. Halevy, P. Norvig, and F. Pereira, "The unreasonable effectiveness of data," *IEEE Intell. Syst.*, vol. 24, no. 2, pp. 8–12, 2009.

[12] Ekin Dogus Cubuk, Ethan S Dyer, Rapha Gontijo Lopes, and Sylvia Smullin, "Tradeoffs in data augmentation: An empirical study," in *ICLR*, 2021. [Online]. Available: https://openreview.net/forum?id=ZcKPWuhG6wy

[13] S. Yang, S. Guo, J. Zhao, and F. Shen, "Investigating the effectiveness of data augmentation from similarity and diversity: An empirical study," *Pattern Recognition*, vol. 148, p. 110204, 2024.

[14] R. Taori, A. Dave, V. Shankar, N. Carlini, B. Recht, and L. Schmidt, "Measuring robustness to natural distribution shifts in image classification." [Online]. Available: http://arxiv.org/pdf/2007.00644v2

[15] D. Hendrycks and T. Dietterich, "Benchmarking neural network robustness to common corruptions and perturbations." [Online]. Available: http://arxiv.org/pdf/1903.12261v1

[16] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 26, no. 1, pp. 43–49, 1978.

[17] E. J. Keogh and M. J. Pazzani, "Relevance feedback retrieval of time series data," in *Proc. of the 22nd Annu. Int. ACM SIGIR Conf. on Res. and Develop. in Inf. Retrieval*, F. Gey, M. Hearst, and R. Tong, Eds. New York, NY, USA: ACM, 1999, pp. 183–190.

[18] Leonard Kaufman and Peter J. Rousseeuw, "Clustering by means of medoids," in *Statistical Data Analysis Based on the L1 Norm and Related Methods*, Yadolah Dodge, Ed., 1987, pp. 405–416.

[19] L. Kaufman and P. J. Rousseeuw, Eds., *Finding Groups in Data*, ser. Wiley Series in Probability and Statistics. Wiley, 1990.

[20] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *J. of Comput. and Appl. Math.*, vol. 20, pp. 53–65, 1987.

[21] D. Feldman, "Core–sets: An updated survey," *WIREs Data Mining and Knowledge Discovery*, vol. 10, no. 1, 2020.

[22] I. Jubran, A. Maalouf, and D. Feldman, "Overview of accurate coresets," *WIREs Data Mining and Knowledge Discovery*, vol. 11, no. 6, 2021.

[23] E. Elhamifar, G. Sapiro, and S. S. Sastry, "Dissimilarity-based sparse subset selection," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 11, pp. 2182–2197, 2016.

[24] R. Yu, S. Liu, and X. Wang, "Dataset distillation: A comprehensive review," *IEEE transactions on pattern analysis and machine intelligence*, vol. 46, no. 1, pp. 150–170, 2024.

[25] H.-P. Kriegel, P. Kröger, and A. Zimek, "Subspace clustering," *WIREs Data Mining and Knowledge Discovery*, vol. 2, no. 4, pp. 351–364, 2012.

[26] Romain Tavenard, Johann Faouzi, Gilles Vandewiele, Felix Divo, Guillaume Androz, Chester Holtz, Marie Payne, Roman Yurchak, Marc Rußwurm, Kushal Kolar, and Eli Woods, "Tslearn, a machine learning toolkit for time series data," 2020.

[27] E. Schubert and P. J. Rousseeuw, "Fast and eager k -medoids clustering: O(k) runtime improvement of the pam, clara, and clarans algorithms," *Inf. Syst.*, vol. 101, p. 101804, 2021.

[28] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization." [Online]. Available: http://arxiv.org/pdf/1412.6980v9

[29] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library."