2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Automatic Loading of Unknown Material with a Wheel Loader Using Reinforcement Learning

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

***Abstract*— Loading multiple different materials with wheel loaders is a challenging task because various materials require different loading techniques. It's, therefore, difficult to find a single controller capable of handling them all. One solution is to use a base controller and fine-tune it for different materials. Reinforcement Learning (RL) automates this process without the need for collecting additional human-annotated data. We investigated the feasibility of this approach using a full-size 24-tonnes wheel loader in the real world and demonstrated that it's possible to fine-tune a neural network controller that was originally trained with imitation learning on blasted rock for use with an unknown gravel material, requiring 20 bucket fillings. Additionally, we showcased the adaptability of a controller pre-trained on woodchips for an unknown gravel material, requiring 40 bucket fillings. We also proposed a novel reward function for the material loading task. Finally, we examined how the sampling time of the reinforcement learning algorithm affects convergence speed and adaptability. Our results demonstrate that it's optimal to match the sampling time of the RL algorithm to the delays of the wheel loader's hydraulic actuators.**

I. INTRODUCTION

Wheel loaders are commonly used in different quarries and construction sites, where one of the most common tasks is pile loading of different materials. This task is repetitive but also complex, and it requires a trained and experienced operator to complete it efficiently in terms of productivity and fuel consumption. It's possible to lower the required operator experience level, increase the productivity and fuel efficiency of the wheel loader, and at the same time reduce the impact of the labor shortage in the construction industry by finding an optimized automated bucket filling assistance system [1]–[3].

Automating the bucket filling task is challenging because the unknown dynamics between the material pile and the machine make it difficult to model and solve with rule-based or optimal control theory [1]. Therefore, the current trend is to use data-driven approaches like imitation learning, as in [4]–[8]. The results show that it's possible to get human-level performance on the bucket filling task for a specific material using imitation learning and Neural Networks (NNs).

Wheel loaders in quarries are usually loading tens of different materials, including sand, gravel, and blasted rock. An automatic system must be able to handle them all, which is another challenge because different materials have different properties such as kernel size, density, and shape, leading to different bucket filling strategies. Besides quarries, wheel loaders are also operated on work sites with completely different materials, such as plastic recycling, bio-waste recycling, woodchips handling. A skilled and experienced human operator learns to automatically adapt the bucket filling strategy depending on the target material. It's shown in [9], [10], that a NN controller trained on one material has degraded performance on a different material. The naive solution is to collect more training data for every new material, which will require significant effort because recording the required data from real machines is expensive. A more effective approach is to use transfer learning between different materials, and it's suggested in [10] that the best strategy is to use a more complex material such as blasted rock as a base controller and then adapt it to other materials. Despite using the data more efficiently, it still requires data collection from each new and unknown material, albeit significantly less than the naive solution.

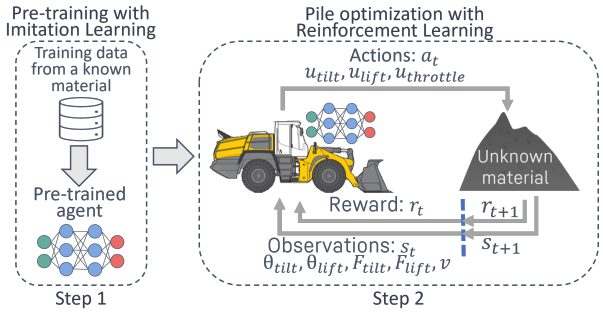


Fig. 1: Overview of the learning process for adapting a controller to an unknown material. The RL agent is first pre-trained on a known material with imitation learning, and then adapted to the unknown material by interacting with it.

Another solution is to use Reinforcement Learning (RL), which is the main focus of this paper, to transfer the controller automatically to the new material without the need for collecting more data from human operators. Instead, the wheel loader explores new behaviors by interacting with the unknown material and optimizing the bucket filling strategy for each scoop. It's possible with RL to optimize a base controller on each of the piles in the quarry automatically without any interaction or input from humans. In order for this to be feasible, the adaptation to the new material must be completed within a reasonable amount of bucket fillings.

*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

This approach has been tested for wheel loaders before in [9], but only with a material similar to the target material.

A problem with RL in the real world is the delay of the actions and observations [11]–[13]. These delays are normally ignored in the standard RL formulation, but they become more apparent in real-world applications and especially for Heavy-Duty Mobile Machines (HDMMs) because of the slow reaction time of the hydraulic actuators. We show in this paper how the sampling time τ_s of the Markov Decision Process (MDP) affects the learning speed and how it's improved by matching it to the delay of the wheel loader's actuators.

The goal of this paper is to show the feasibility of using a pre-trained neural network controller synthesized with imitation learning and transferred to an unknown material with RL, as shown in Fig. 1. We formulate a novel reward function for the bucket filling task and show that it's possible to adapt a controller pre-trained on blasted rock to a gravel pile after 20 episodes, as well as a controller pre-trained on woodchips after 40 episodes.

II. RELATED WORK

Many approaches for automatic bucket filling have been tested in the past, some researchers have optimized the bucket filling trajectory, and they show in [14]–[16] that it's possible to increase fuel efficiency with satisfying fill-grade and loading times in a simulator. However, we usually don't know all the material characteristics in a real-world test.

Data-driven approaches have also been researched in the past for the bucket filling problem, which has yielded satisfying results for loading a single material [4]–[9]. We investigated a transfer learning approach for loading multiple materials in our previous paper, which made it possible to load three distinct kinds of materials: sand, gravel, and blasted rock. The controllers were pre-trained on blasted rock, and we then used 10 additional samples for fine-tuning the network for the other materials [10].

RL is a promising option for transfer learning that doesn't require any further human-recorded samples. RL has, for example, been used for robotic arm manipulation [17], to teach a four-legged robot to walk [18]–[20], as well as different kinds of HDMMs. The researcher in [21] uses the Proximal Policy Optimization (PPO) algorithm to learn a forestry crane log grasping with a success rate of 97% in the simulator. PPO was also used in [22], [23], to automate the digging of an excavator capable of adapting to different soil conditions within a single scoop. They pre-trained the agent in a simulator and then fine-tuned the control policy for the real machine.

RL has also been applied previously to wheel loaders, both in simulation and the real world, for miniature loaders and for full-scale machines. The authors of [24] used the Soft Actor Critic (SAC) algorithm to train an underground loader in a simulated environment. Another researcher developed a loading strategy first in the simulator and then transferred the policy to a real machine, but the machine was only a toy model with significantly different dynamics from a full-size machine used in quarries and construction sites. Furthermore, they only trained on one material and calculated the reward function from the trajectory through the pile, given by sensors unavailable to us [25]. There has also been research on predicting the reward function for a small wheel loader in [26].

The previous work most closely related to ours is from the researchers in [9]. They adapt a time-delayed neural network trained on medium coarse gravel, with particle sizes up to 64 mm, to a pile of gravel cobble, with a particle size up to 200 mm, for a full-size 20-tonnes wheel loader in the real world. They use the Deterministic Policy Gradient (DPG) algorithm, but with an action space of only 2 degrees of freedom. The agent is in control of the lift and tilt actions, while the throttle value is kept constant. Our implementation instead uses an action space with 3 dimensions, which includes the throttle. Furthermore, they are not pre-training the critic, while we pre-train it with the training data. They are also not using any exploration noise, which limits the exploration of the agent, potentially slowing the learning speed. Another difference is the reward function, they use a reward function based only on the lift force F_{lift}, while our reward function is based on the work done by the hydraulic cylinders. They reported an increase in the material weight of about 700 kg and a decrease in the loading time of 0.6 s after 40 bucket fillings when using a pre-trained controller from a similar material. We improve upon the previous work by adapting faster to a new pile, as well as demonstrating that it's possible to adapt to a material that is very different from the training material.

III. METHOD

Our method consists of first collecting training data with an expert operator from one material and create a NN controller suitable to load that material with imitation learning. We then use reinforcement learning to optimize and transfer the controller to an unknown material not seen in the training data by fine-tuning the last layer of the neural network controller as in Fig. 1.

A. Bucket Filling algorithm

We use the same bucket filling algorithm as in our previous work [8], [10] which is inspired by [5]. Fig. 2 shows a summary of this approach and the three different phases: approach pile, loading, and exit.

During the approach phase, the bucket is placed flat on the ground, and the throttle is activated by 50% to make the wheel loader drive automatically to the pile. When the force in the lift cylinder reaches a certain threshold, the NN controller is activated to produce lift, tilt, and throttle

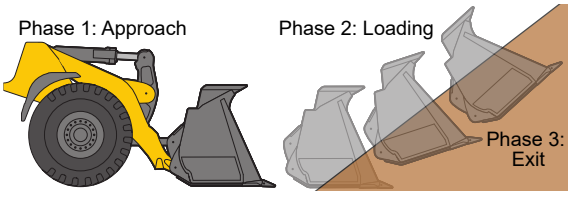


Fig. 2: The three phases of the bucket filling algorithm.

commands. The NN is capable of using the full range of the working hydraulics, meaning it can both tilt in and tilt out the bucket, and both lift and lower the boom. This phase ends when the bucket and boom have reached the end position, then the exit phase has started and the bucket is fully tilted in. As a last step, the material in the bucket is weighted to give the terminal reward to the agent.

B. Imitation learning

The neural network controllers use the same network architecture and training procedure as in our previous work [8], [10]. We showed in [8] that the CNN architecture used here had the best robustness and performance in terms of fill-grade and loading time. The architecture of the neural network controller is shown in Fig. 3 and has 5 input signals: the bucket and boom angles θ_{tilt}, θ_{lift}, forces in the tilt and lift cylinders F_{tilt}, F_{lift}, and the machine velocity v as shown in Fig. 4. Each input signal is sampled every 15 ms, and a stack of 16 samples is sent as input to the network in order to capture temporal patterns in the data. The first layer consists of a 1-Dimensional Convolutional Neural Network (1D-CNN) with 3 filters and a kernel size of 5. The output of the CNN is then flattened and connected to a fully connected layer with a hidden size of 64 and 3 output nodes, one for each of the control commands: u_{tilt}, u_{lift}, and $u_{throttle}$. Each layer of the neural network is activated by the Rectified Linear Unit (ReLU) except for the last which is activated by the hyperbolic tan (Tanh) function since the output of the network should be in the range of $[-1, 1]$ to control the full-range hydraulic actuators.

The NNs are trained using supervised learning to predict the operator commands given the 5 input signals with respect to the mean square error between the predicted and true values using the ADAM optimizer [27]. The neural network is implemented using PyTorch [28].

C. Reinforcement learning

Tasks in reinforcement learning are usually modeled as a Markov Decision Process (MDP) with discrete time steps, t, sampled with an interval, τ_s. An MDP is defined by a state space, $s_t \in \mathcal{S}$, an action space, $a_t \in \mathcal{A}$, a transition dynamics probability function, $p(s'|s, a)$, and a reward function $r_t = r(s_t, a_t)$.

The goal of RL is to find the optimal policy $\pi_*(s_t, a_t)$ that maximizes the episodic reward $R_t = \sum_{i=t}^{T} \gamma^{(i-t)} r(s_t, a_t)$ with the discount factor γ and the terminal state T.

We implemented the Proximal Policy Optimization (PPO) algorithm with generalized advantage estimation [29], [30]. Our implementation closely follows the implementation from the open source library, Stable-Baselines3 [31]. PPO has been used with success for excavators and log handlers [21]–[23], and it also requires little hyperparameter tuning.

The PPO algorithm is an on-policy actor-critic algorithm, which means that it updates the same policy used for sampling actions, and it uses one network to parameterize the policy π and one network to estimate the value of the current state s_t. We start with a copy of the pre-trained NN

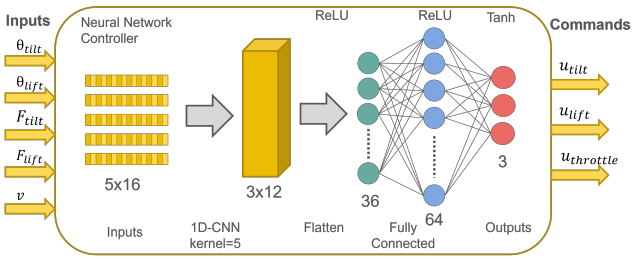


Fig. 3: Architecture of the NN trained with imitation learning as well as the actor network. The NN receives the joint angles, forces, and velocity.

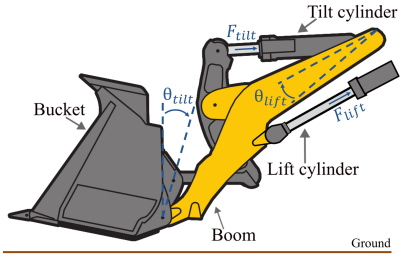


Fig. 4: Definition of the joint angles and forces.

as the actor and fine-tune the last layer after each episode. The critic network is also a copy of the pre-trained network, but with 1 output neuron instead of 3, since it only predicts the value of the current state. The last layer of the critic network is initialized with random weights.

We copy the pre-trained agent for both the actor and the critic because the first layer of the pre-trained NN controller has a CNN that finds common patterns in the input signals and is thus general and useful for both the actor and the critic.

The standard PPO algorithm has a fixed rollout size, which means the actor and critics might get updated while the wheel loader digs through the pile. However, updating the networks during digging resulted in jerky motions, we therefore limited the updates to after each episode.

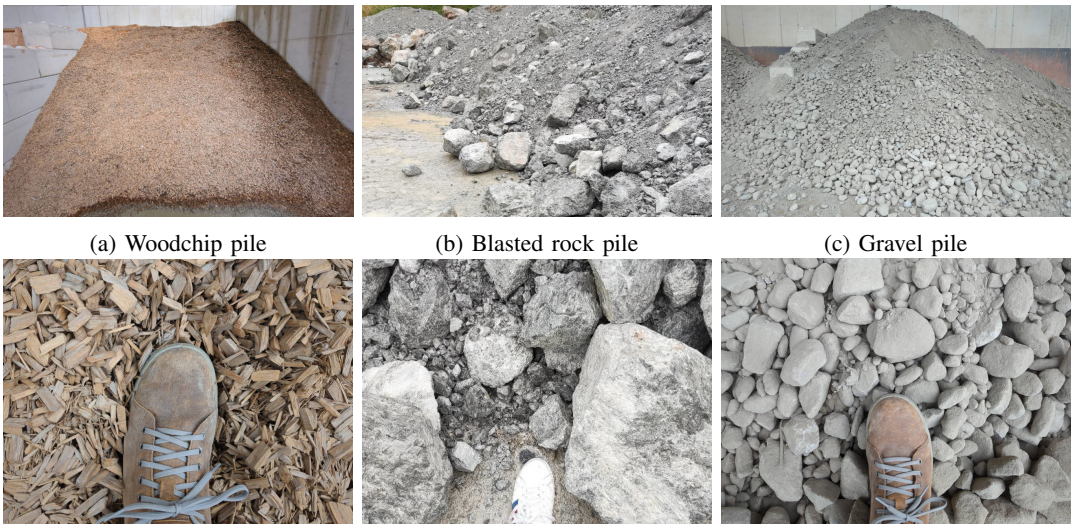
We add a learnable noise parameter to promote exploration during each bucket filling. The noise is modeled as a normal distribution $\mathcal{N}(\mu_{noise}, \sigma_{noise})$ and added to the actions.

D. Reward function

The reward function plays an important role in determining the performance and learning speed of the RL agent, and it should reflect what we want to achieve and not how to achieve it. A dense reward function where the agent receives a non-zero reward in each time-step is also important for effective learning.

We formulate the reward function based on the work done by the hydraulic cylinders and the final material weight in the bucket. The idea is that if the cylinders carry out a higher amount of work during loading, the final bucket weight will also be high.

Equation (1) is the step reward and is given to the agent in each time-step t except for the terminal $t = T$, where the agent receives the terminal reward r_T described in (2).



(a) Woodchip pile (b) Blasted rock pile (c) Gravel pile

(d) Close up of the woodchip pile (e) Close up of the blasted rock pile (f) Close up of the gravel pile

Fig. 5: The three different materials used in the experiments. The woodchip and blasted rock were used for training the pre-trained network, while the gravel pile was used for evaluation of the RL optimization.

$$r_t(s, \mathbf{w}) = \tanh(w_1 v_{tilt} F_{tilt} + w_2 v_{lift} F_{lift}) - 1 \quad (1)$$

The variables v_{tilt} and v_{lift} are the speeds of the tilt and lift cylinders respectively. Each component of the reward can be scaled with the weights w_1 and w_2. We use the Tanh function to restrict the calculated work to $[-1, 1]$ and the negative term, -1, to encourage the agent to finish the episode as soon as possible. Without any time penalty, the agent will probably try to accumulate as much reward as possible and therefore never end the episode.

The terminal reward is given below by (2).

$$r_T(s) = \begin{cases} w_3 m & \text{if successful} \\ -1 & \text{if bucket filling unsuccessful} \end{cases} \quad (2)$$

The terminal reward is calculated from the loaded mass m [kg], which is available at the end of an episode if the agent was successful in completing the bucket filling. The mass is then scaled by the factor w_3. A successful bucket filling is defined as the controller reaching the end position of the bucket without getting stuck or being manually aborted by the safety driver. The bucket might get stuck in the pile if there is too much material above it, resulting in the stalling of either the lift or tilt cylinder. Another error mode is excessive wheel spin, which occurs if the agent uses too much throttle relative to the friction between the ground and the wheels.

IV. Experimental setup

This section describes the training data, target material, and the wheel loader setup used for the experiments in section V.

A. Training data and target material

We collected data from woodchips and blasted rock, each dataset consisting of 100 bucket fillings. The data were collected by a professional wheel loader operator, and the same setup was used for data collection as the experiments, with one exception. We used a bigger bucket when collecting the woodchips data because of its low density properties. In both cases, the data consists of the sensor signals from the machine and the operator's joystick and throttle commands.

Fig. 6: The wheel loader and the target material used for the experiments.

The woodchip data and the blasted rock data were used in the first step in Fig. 1 for pre-training two different agents. A third material, a type of gravel, was used as the target material, i.e. the unknown material to optimize the bucket filling for. All three materials are shown in Fig 5 with an overview image of the pile as well as a close-up of the material using a foot as a scale.

We chose blasted rock because it's better to use a complex material for pre-training and then fine-tune the model to the target material [10]. We chose woodchips because the material properties, such as shape and density, are very different from the target gravel material, thereby illustrating the learning capabilities of our process.

B. Wheel loader setup

We evaluate our system on a full-size wheel loader, specifically a 24-tonnes Liebherr 576, and a consumer PC for running the RL algorithms using the ROS2 framework [32]. The wheel loader and the gravel pile used in the experiments is shown in Fig. 6. The PC communicates with the wheel

loader over the CAN-bus, and the wheel loader uses only the standard sensors available on the default factory machine without any modifications. The sensors measure joint angles (θ_{tilft}, θ_{lift}), cylinder forces (F_{tilt}, F_{lift}), and the machine velocity v as shown in Fig. 4.

The terminal reward is capped at 10 000 kg which represents a full bucket with the setup used for the experiments.

V. EXPERIMENTS

We test the learning speed and performance of our approach by investigating how well the agent adapts the bucket filling behavior to an unknown material not seen in the training data used for pre-training. We also empirically show how the sampling time τ_s of the MDP can be tuned for slow-reacting hydraulic machines to improve the convergence speed of the reinforcement learning.

A. Sampling time

The time from sending the command to the wheel loader until the actuators move is not instantaneous. Therefore, we first measure the action time delay of the wheel loader's hydraulic system and then test how different sampling times, τ_s, affects the convergence speed of the PPO algorithm. The sampling time, τ_s, changes only the sampling time of the actions and not the sampling time of the input data. In other words, the neural network receives a stack of 16 samples sampled at 15 ms, but the action is only updated every τ_s ms.

The action delay was found by lifting the boom with 100% lift command from steady state and measuring the time between sending the command and when the boom started raising. We found that it was approximately 240 ms, as seen in Fig. 7.

We investigated 4 different values of τ_s: 15 ms, 120 ms, 240 ms, and 480 ms, and tested how quickly the agent could increase the accumulated rewards per episode. By changing τ_s we also change the number of steps the PPO algorithm takes and thereby the accumulated step rewards. In order to make the tests comparable, we have normalized the step reward with τ_s. The results in Fig. 8 show the agent is learning the fastest with $\tau_s = 240$ ms, and the learning speed increases as the sampling time increases. However, choosing τ_s too high results in the agent getting stuck in the pile more often, which is seen in the negative values of the accumulated reward. One reason could be that the agent doesn't have time to react quickly enough to changes in the input.

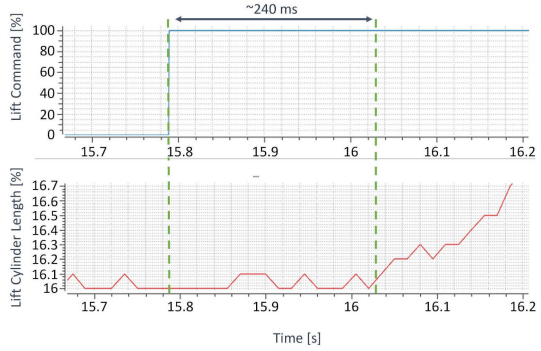


Fig. 7: Delay between sending the lift command and observable movements of the hydraulic lift cylinder.

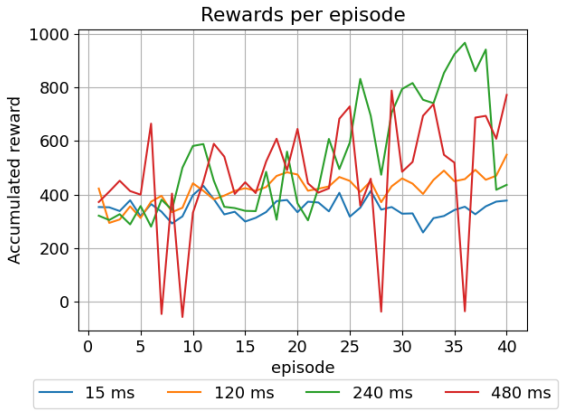


Fig. 8: The accumulated rewards for each episode for different sampling times τ_s.

A low value of τ_s on the other hand, makes it hard for the PPO algorithm to associate the right actions with the right reward and state because of the delay in the hydraulic actuators observed in Fig. 7. Another reason might be that the actuators do not have time to react to the commands if the action is switched frequently, making it harder for the agent to assign the correct values to the visited states. This problem is amplified with action noise because it's more likely that the next action is further away from the last one, thus changing the action too rapidly for the actuators to react, results in no or jerky motions.

B. Unknown pile optimization

For the next experiment, we optimized two different pre-trained networks on the gravel pile shown in Fig. 5, one pre-trained on woodchips and the other on blasted rock. We used $\tau_s = 240$ ms for these experiments, as indicated by the previous experiment, and the PPO hyperparameters from table I.

We ran RL for 20 and 50 episodes for blasted rock and woodchips respectively, and evaluated the bucket filling performance after every 10th episode by running the policy without noise or any gradient updates for 10 bucket fillings. We saw from early experiments that the sampling time τ_s only affects the RL training and has little effect on the bucket filling performance for the pre-trained agents, which are not using any action noise. Therefore, the command signals were updated with 15 ms intervals instead of 240 ms to be more consistent with the controllers not using RL during

TABLE I: PPO Hyperparameters

γ	0.99	entropy coefficient	0.01
σ_{noise}	0.37	learning rate α	0.00025
μ_{noise}	0	update epochs	10
λ_{gae}	0.95	max grad norm	0.5
w_1	0.5	clip coefficient	0.2
w_2	0.5	value function coefficient	0.5
w_3	8000		

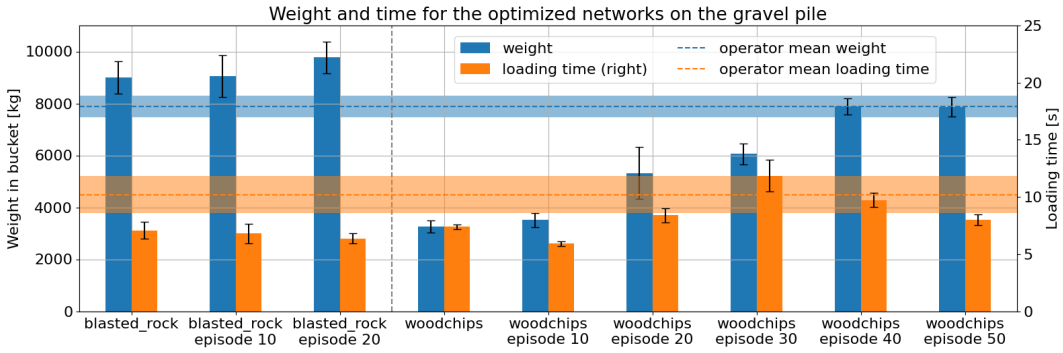


Fig. 9: The mean loading time (orange) and the mean material weight (blue) for the agents after every 10th episode, with the corresponding standard deviation. 10 bucket fillings per controller were used for the evaluation, and the horizontal line represents the performance of the human operator. The agents left of the gray dashed line used blasted rock data for pre-training, and the agents right of the line used woodchips for pre-training.

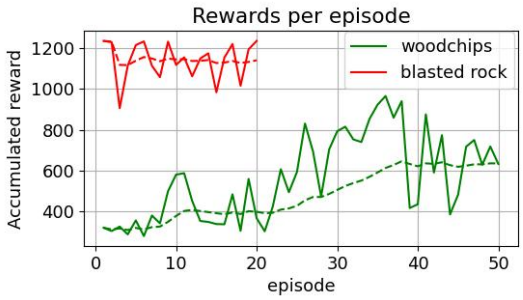


Fig. 10: The green line shows the agent pre-trained on woodchips and the red line on blasted rock. The dotted lines are the average rewards.

the performance evaluation. The performance was evaluated based on the material weight in the bucket as well as the loading time. The loading time is defined from the moment the bucket penetrates the pile until the stopping criteria are met.

A comparison of the agents' performance to each other and to a human operator baseline is shown in Fig. 9, and the accumulated rewards for each episode are shown in Fig. 10. The learning progress is also visualized in the accompanying video material.

The results show that the agent pre-trained on woodchips loads only 3000 kg of material before the RL optimization, but it's able to adapt to the gravel material after 40 episodes, which is about 40 min of real-world training. The agent achieves comparable results to our human operator baseline after the adaptation, and we can also see that the loading time is further improved after 50 bucket fillings.

When comparing the pre-trained woodchips agent to the optimized network after 40 episodes, we observe that the agent is learning to use a higher throttle command and thus gets deeper into the pile. A higher throttle value when loading gravel compared to woodchips is necessary because of the higher density of the material. However, it's still not performing at the same level as the agent trained on blasted rock, which might be solved by more training.

We also show that it's possible to further improve an already well-performing pre-trained agent when using blasted rock data. We improved the mean bucket weight from 9000 kg to 9700 kg, and the mean loading time from 7.0 s to 6.4 s. These results confirm our previous result in [10] that the best strategy is to pre-train on a complex material and optimize the network for the target material, either via transfer learning or reinforcement learning.

VI. CONCLUSION

We demonstrate the feasibility of pre-training a neural network controller on a complex material such as blasted rock and use RL to fine-tune the controller to an unknown gravel material with 20 episodes and with 40 episodes when pre-training on woodchips. We also demonstrate how adapting the sample time of the PPO algorithm to the delay of the actuators can improve the learning speed. However, more investigations are necessary to find out why the convergence speed is improved by this.

Furthermore, we show that it's feasible to use a complex material such as blasted rock and adapt it to a gravel material. These results suggest that it's possible to run the RL optimization in the background and constantly adapt to the different materials that the machine is encountering.

The adaptation of the woodchips material can be further improved by optimizing the reward function and exploration strategy. More research is necessary to find a reward function that is more reflective of the goal and an exploration strategy that can better exploit the underlying mechanics of the bucket filling problem. Furthermore, more investigation should be conducted in RL with action and observation delays so that the sampling time can be reduced, thus increasing the number of steps taken during an episode, which means that more states in the observation space are visited.

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

[2] 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.

[3] B. Frank, L. Skogh, R. Filla, A. Froberg, and M. Alaküla, "On increasing fuel efficiency by operator assistance systems in a wheel loader."

[4] S. Dadhich, U. Bodin, F. Sandin, and U. Andersson, "Machine learning approach to automatic bucket loading," in *2016 24th Mediterranean Conference on Control and Automation (MED)*. IEEE, 21/06/2016 - 24/06/2016, 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," *Automation 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 International Conference on Robotics and Automation (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 International Conference on Robotics and Automation (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," *Automation in Construction*, vol. 150, p. 104843, 2023.

[9] 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 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 19/07/2020 - 24/07/2020, pp. 1–9.

[10] D. Eriksson, R. Ghabcheloo, and M. Geimer, "Towards multiple material loading for wheel loaders using transfer learning," in *Proceedings of 18th Scandinavian International Conference on Fluid Power, SICFP23*, 2023.

[11] K. V. Katsikopoulos and S. E. Engelbrecht, "Markov decision processes with delays and asynchronous cost collection," *IEEE Transactions on Automatic Control*, vol. 48, no. 4, pp. 568–574, 2003.

[12] E. Schuitema, L. Busoniu, R. Babuska, and P. Jonker, "Control delay in reinforcement learning for real-time dynamic systems: A memoryless approach," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2010, pp. 3226–3231.

[13] S. Ramstedt, Y. Bouteiller, G. Beltrame, C. Pal, and J. Binas, "Reinforcement learning with random delays." [Online]. Available: http://arxiv.org/pdf/2010.02966v3

[14] 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*, ser. Linköping Electronic Conference Proceedings. Linköping University Electronic Press, 2017, pp. 402–417.

[15] B. Frank, J. Kleinert, and R. Filla, "Optimal control of wheel loader actuators in gravel applications," *Automation in Construction*, vol. 91, pp. 1–14, 2018.

[16] J. Yao, C. P. Edson, S. Yu, G. Zhao, Z. Sun, X. Song, and K. A. Stelson, "Bucket loading trajectory optimization for the automated wheel loader," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 6, pp. 6948–6958, 2023.

[17] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 29/05/2017 - 03/06/2017, pp. 3389–3396.

[18] L. Smith, J. C. Kew, X. Bin Peng, S. Ha, J. Tan, and S. Levine, "Legged robots that keep on learning: Fine-tuning locomotion policies in the real world," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 1593–1599.

[19] L. Smith, I. Kostrikov, and S. Levine, "A walk in the park: Learning to walk in 20 minutes with model-free reinforcement learning." [Online]. Available: http://arxiv.org/pdf/2208.07860v1

[20] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to walk in minutes using massively parallel deep reinforcement learning," in *Proceedings of the 5th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, A. Faust, D. Hsu, and G. Neumann, Eds., vol. 164. PMLR, 2022, pp. 91–100. [Online]. Available: https://proceedings.mlr.press/v164/rudin22a.html

[21] J. Andersson, K. Bodin, D. Lindmark, M. Servin, and E. Wallin, "Reinforcement learning control of a forestry crane manipulator," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 2121–2126.

[22] P. Egli and M. Hutter, "A general approach for the automation of hydraulic excavator arms using reinforcement learning," 2021.

[23] P. Egli, D. Gaschen, S. Kerscher, D. Jud, and M. Hutter, "Soil-adaptive excavation using reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 9778–9785, 2022.

[24] S. Backman, D. Lindmark, K. Bodin, M. Servin, J. Mörk, and H. Löfgren, "Continuous control of an underground loader using deep reinforcement learning," *Machines*, vol. 9, no. 10, p. 216, 2021.

[25] O. Azulay and A. Shapiro, "Wheel loader scooping controller using deep reinforcement learning," *IEEE Access*, vol. 9, pp. 24 145–24 154, 2021.

[26] N. Strokina, W. Yang, J. Pajarinen, N. Serbenyuk, J. Kämäräinen, and R. Ghabcheloo, "Visual rewards from observation for sequential tasks: Autonomous pile loading," *Frontiers in Robotics and AI*, vol. 9, 2022.

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

[28] 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," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d\textquotesingle Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc, 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[29] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms." [Online]. Available: http://arxiv.org/pdf/1707.06347v2

[30] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation." [Online]. Available: http://arxiv.org/pdf/1506.02438v6

[31] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021. [Online]. Available: http://jmlr.org/papers/v22/20-1364.html

[32] Steven Macenski, Tully Foote, Brian Gerkey, Chris Lalancette, and William Woodall, "Robot operating system 2: Design, architecture, and uses in the wild," *Science robotics*, vol. 7, no. 66, p. eabm6074, 2022.