Adaptation of a wheel loader automatic bucket filling neural network using reinforcement learning

Siddharth Dadhich, **Fredrik Sandin**, **Ulf Bodin**, **Ulf Andersson**
Luleå University of Technology, 97187 Luleå, Sweden; ulf.bodin@ltu.se

Torbjörn Martinsson
Volvo CE, Bolindervägen 5, 63185, Eskilstuna, Sweden.

***Abstract*—Bucket-filling is a repetitive task in earth-moving operations with wheel-loaders, which needs to be automated to enable efficient remote control and autonomous operation. Ideally, an automated bucket-filling solution should work for different machine-pile environments, with a minimum of manual retraining. It has been shown that for a given machine-pile environment, a time-delay neural network can efficiently fill the bucket after imitation-based learning from 100 examples by one expert operator. Can such a bucket-filling network be automatically adapted to different machine-pile environments without further imitation learning by optimization of a utility or reward function? This paper investigates the use of a deterministic actor-critic reinforcement learning algorithm for automatic adaptation of a neural network in a new pile environment. The algorithm is used to automatically adapt a bucket-filling network for medium coarse gravel to a cobble-gravel pile environment. The experiments presented are performed with a Volvo L180H wheel-loader in a real-world setting. We found that the bucket-weights in the novel pile environment can improve by five to ten percent within one hour of reinforcement learning with less than 40 bucket-filling trials. This result was obtained after investigating two different reward functions motivated by domain knowledge.**

***Index Terms*—reinforcement learning, imitation learning, bucket filling, wheel loader, automation, construction.**

I. Introduction

Bucket filling is a repetitive task for wheel-loader and excavator operators, making it suitable for automation. In particular, bucket-filling automation is required for efficient remote control, for example in hazardous environments, and for the development of fully autonomous solutions. There is no general automatic bucket-filling solution for different machine-pile environments. Caterpillar and Komatsu provide semi-autonomous loading functions, but only for limited applications (soil and medium fine gravel) with loose material. These solutions are not robust and frequently fail in different conditions [1]. Wheel-loaders are versatile machines often used as multi-purpose machines at production sites [2]. A bucket-filling solution that functions in different environments and scenarios, with changing machine and material properties, is therefore desired. This paper investigates an algorithm capable of adapting a bucket-filling model for a wheel-loader to a different environment, with a limited amount of data. Our main contribution is the demonstration of an imitation and reinforcement based learning approach for training and adaptation of a bucket-filling neural network, with realistic experiments where the network operates in closed loop.

This work is funded by the program "Fordonsstrategisk Forskning och Innovation, FFI", Vinnova grant number 2017-01958. FS contribution is funded by The Kempe Foundations under contract SMK-1429.

The automatic bucket-filling problem is challenging, which is evident from the fact that there are no fully autonomous solutions [3] for different machine-pile environments. Previous approaches to automate bucket-filling can be summarized into three categories: 1) trajectory based, 2) compliance control and 3) rule based logic. The trajectory based approaches [4]–[6] aim to study and reproduce efficient bucket trajectories of expert operators. The compliance control approach aims to sense the interaction forces between the pile and machine [7] and act dynamically to either adjust a preplanned trajectory [8], [9] or regulate a target force on pistons [10]. The rule based approaches [11], [12] have employed fuzzy logic to condense expert operator data into simpler rules for producing primitive bucket motion commands.

The previous approaches have demonstrated successful results to varying degrees. However, these results have being shown in specific machine-pile environments. For example, the admittance control based autonomous bucket-filling algorithm presented in [10], [13] is developed for load-haul-dump machines and cannot be easily used for wheel-loaders, which are more common. The previous solutions to automate this task do not generalize to different machines and pile environments. Therefore, there is a need of a generic automatic bucket-filling solution that can be easily adapted to different scenarios.

Reinforcement learning (RL) is a framework for the most general type of learning, where an RL-agent learns to maximize expected cumulative rewards by interacting with its environment, without explicit programming [14]. In recent breakthroughs in RL, deep RL, i.e., the use of neural networks in RL, is shown to play Atari games [15], [16], chess and Go [17] at or above human performance.

Kober et al. [18] reviews the use of RL in robotics and highlights the problems with using RL for real-time control and presents different ways to mitigate them. Given the curse of dimensionality problem in RL, function approximation becomes inevitable for continuous domain problems [18]. Policy gradient algorithms are widely used in problems with continuous actions. Silver et al. [19] presents a deterministic actor-critic algorithm based on policy gradient similar to

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

Fig. 1. Volvo L180H wheel-loader in manual operation.

NFQCA (Neural Fitted Q Iteration with Continuous Actions) [20]. Similar to our work, a real-world application of RL is driving in a lane [21], which uses DDPG (Deep Deterministic Policy Gradient) [22], which is also an adaptation of the deterministic actor-critic algorithm.

We employ the deterministic policy gradient (DPG) actor-critic algorithm from [19] to adapt and improve a baseline imitation learning based bucket-filling model, proposed in [23]. The core of the imitation model is a time-delayed neural network which predicts lift and tilt joysticks actions during the bucket-filling. The imitation learning model is successful in bucket-filling after an initial period of supervised training with 100 examples from an expert operator, without using any information about the material or the pile.

The imitation learning bucket-filling solution is developed in a specific environment. This solution is trained, tested and validated on one material which is medium coarse gravel (fine particles up to 64 mm in size) and one machine, a Volvo L180H wheel-loader, see Fig. 1. Although this method does not assume any information from the machine or the pile environment, the performance of this model tuned in one specific setting cannot be guaranteed with changes in operating conditions. For example, a bucket-filling model fine tuned for performance on a machine may under-perform if the same machine is operated with a different bucket. Similarly, changes in the properties of piles, like the size distribution and wetness may lead to under-performance of the bucket-filling algorithm.

One possibility to adapt the imitation model for different conditions is to retrain the underlining time-delayed neural network model. This will require to collect new data for notable changes in operating conditions. Although collecting 100 bucket-filling examples takes only an hour of operation, this can still be too costly for a company in production. If operating conditions are constantly changing, retraining the model for each different condition may be infeasible. Another method is to use transfer learning [24], in which a model trained for a specific condition can be used for a different condition by retraining only the layers closer to the output with a much smaller dataset.

In this paper, we propose an RL based approach to adapt imitation based bucket-filling models for changing conditions. In this approach, the model (an RL agent) receives reward (a reinforcement signal), at every time step until the end of the bucket-filling. The goal of the RL-agent is to maximize the expected sum of rewards. In a traditional RL setup, the agent starts from blank taking random actions in its environment. In a complex task such as the bucket-filling, it can be assumed that taking random actions (lift and tilt joystick actuations) cannot produce a successful bucket-fill. Here, the RL-agent starts from an imitation based bucket-filling model trained and validated in an old setting. We examine the adaption of the model from medium coarse gravel (upto 64 mm particles in size), which is the old setting to a gravel-cobble pile (upto 200 mm particles in size), the new setting.

We present results with the deterministic actor-critic algorithm adapting an imitation model trained in medium coarse gravel to a cobble-gravel pile. The experiments presented in this paper are performed using the same experimental machine, a Volvo L180H wheel-loader. The employed RL algorithm is model-free and on-policy, and configured with a goal to increase productivity of the bucket-filling task. Experiments show an improvement in productivity of bucket-fillings in less than 40 examples.

II. Background

A. Imitation learning based bucket-filling algorithm

Imitation learning is a sequential process where a machine learning model is trained to predict the actions of an expert operator that successfully fills the bucket in a particular pile environment, such that the resulting model can mimic the operator with good performance. The analysis of the wheel-loader data shows that the loading process can be divided in four phases, as shown in Fig. 2. In the first phase, the operator maintains a constant speed of the machine when approaching the pile. The onset of the second phase is defined by a sharp increase in the lift joystick signal, which increases the pressure on the front tires in order to avoid wheel-spin resulting in increased wear and maintenance costs [25]. In the third phase, the operator modulates the lift, tilt and throttle actions simultaneously to navigate the bucket through the pile and obtain a high bucket-weight in a short period of time. The fourth phase indicates the finishing part of the bucket-filling process and involves tilting the bucket until breakout, i.e., when the bucket exits the pile.

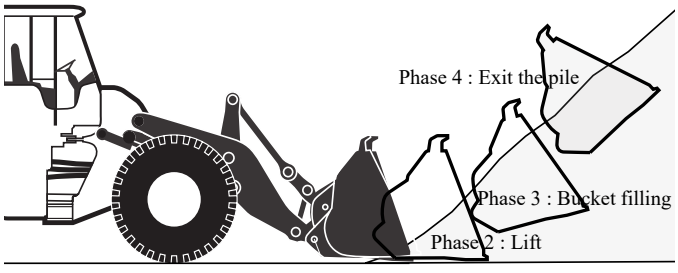

Fig. 2. Different phases of the loading process implemented in the imitation learning based automatic bucket filling algorithm.

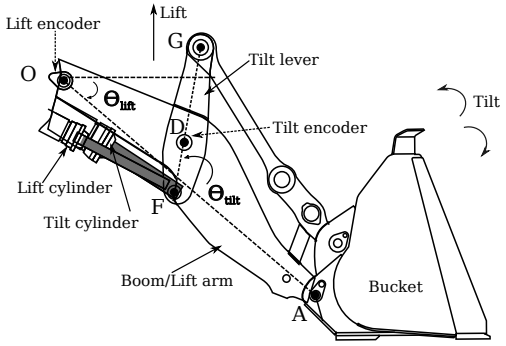

Fig. 3. Bucket linkage for Volvo L180H wheel-loader highlighting the definition of the lift and tilt angles, and the position of the angle encoders.

During each phase, the imitation based bucket-filling algorithm uses different constant values for the throttle. In phase three, the imitation model predicts joystick control commands for the lift and tilt actions based on signals that are easily acquired from the machine. A hardware interface to the engine control units (ECUs) of the experimental machine, a Volvo L180, is used to access machine data (such as the lift and tilt angle, and angular velocities and drive axle speed) and also to automatically control the machine. The bucket linkage diagram of the Volvo L180 in Fig. 3 illustrates the lift and tilt cylinders, and the definitions of the lift and tilt angles. Additionally, the experimental wheel-loader is instrumented with pressure transducers at each end of the lift and tilt cylinders, which are used to measure the forces applied on the lift and tilt pistons. The imitation learning model uses forces on the lift and tilt pistons, lift and tilt angles, angular velocities and the speed of the drive axle as input features for prediction of the lift and tilt joystick commands.

1) Time delayed neural network model: Previous works [26], [27] motivate the use of non-linear models to automate bucket-filling. A time-delayed neural network, originally proposed in [28], incorporates a finite memory of previous inputs using a transparent feed-forward architecture. The time-delayed neural network model for bucket-filling (shown in Fig. 4) is proposed in [29] and the corresponding solution is tested and validated in [23] on medium coarse gravel. The time-delayed neural network model for automatic bucket-filling via imitation learning is a three layer network that uses current and past values of features at the input layer. Instead of a regular hidden layer, the middle layer performs action classification to improve the performance and correspondence between the regression model output and the actions performed by an expert operator.

2) Regression by classification: The middle layer in the time-delayed neural network in Fig. 4 is connected to discretized values of the operator's lift and tilt joystick actions, divided into six classes. The class output layer has a softmax output unit and consequently produces a probability distribution over six actuation levels. This design is motivated by the fact that operator actions appear to cluster at different levels [27]. The output layer then combines the probability outputs of the middle layer to produce continuous actions in a regression by classification sense, introduced in [30]. The output layer has a rectified linear activation that only allows positive action values, which is motivated by the fact that bucket-filling in medium coarse gravel only requires unidirectional joystick actions (curling-in and lifting-up of the bucket).

B. Reinforcement learning

Reinforcement learning (RL) is a branch of machine learning which deals with learning from interactions with an environment. Sutton and Barto [31] introduces RL in a Markov Decision Problem (MDP) framework, which consists of an agent exploring a state-space (the RL agent's environment) via taking actions and obtaining rewards for intended behavior.

The main elements of an MDP are a state-space $\mathcal{S}$, an action-space $\mathcal{A}$, a transition probability distribution $p : \mathcal{S} \times \mathcal{A} \to \mathcal{P}(\mathcal{S})$ and a reward signal $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. The agent interacts with its environment using a policy which can be stochastic $\pi_\theta : \mathcal{S} \to \mathcal{P}(\mathcal{A})$ or deterministic $\pi_\theta : \mathcal{S} \to \mathcal{A}$. These interactions generate trajectories of states, actions and reward (s_1, a_1, r_1), (s_2, a_2, r_2), ... (s_T, a_T, r_T), until one of the terminal states is reached. A discount factor $\gamma \ \epsilon \ [0, 1]$ defines a trade-off between current rewards and future rewards. The return R_t^γ is the total discounted reward from time t onwards, $R_t^\gamma = \sum_{k=t}^{T} \gamma^{k-t} r(s_k, a_k)$. This implies that, if $\gamma = 0$ the return is the current reward, in contrast to $\gamma = 1$ which results in a return that is the total sum of rewards from time-step t. The state-value function $V^\pi(s)$ is the expected return after starting from a state s and following a policy π, $V^\pi(s) = \mathbb{E}[R_1^\gamma | s, \pi]$, while the action-value function $Q^\pi(s, a)$ is defined as the expected return after staring from a state s, taking an action a, and following a policy π thereafter, $Q^\pi(s, a) = \mathbb{E}[R_1^\gamma | s, a, \pi]$.

The goal of the reinforcement learning process is to learn from experiences to obtain a policy that maximizes the cumulative discounted reward from the starting state, i.e. the performance objective $J(\pi) = \mathbb{E}[R_1^\gamma | \pi]$. In model based RL the aim is to estimate the transition probability distribution, which is then used in planning to update the value function or

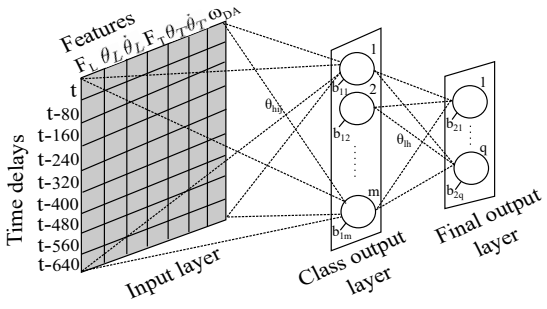

Fig. 4. A time-delayed neural network model used to perform automatic bucket-filling. The input layer uses time-delayed values of features with a delay step of 80 ms and a total delay time of 640 ms. The class output layer has 12 neurons, six each for the lift and tilt categorical predictions. The final output layer combines softmax activations of the previous layer into continuous predictions for the lift and tilt joystick actions.

policy. In model-free RL the value or policy is updated directly without the need of model. The Bellman equations are used to decompose the value functions in a recurrent form. The Bellman equation for the action-value function is

$$Q^{\pi}(s_t, a_t) = \mathbb{E}[r(s_t, a_t) + \gamma Q^{\pi}(s_{t+1}, a_{t+1})]. \quad (1)$$

The error of the Bellman equation is called the temporal difference (TD) error. TD learning, which aims to minimize the TD error, is a central concept in reinforcement learning [31].

Value iteration and policy iteration are two different approaches in model-free reinforcement learning. The value iteration approach aims to solve the Bellman equation to obtain an optimal value function $Q^*(s, a)$ and derive a policy directly from the optimized value function. Value iteration methods suffer significantly more from the curse of dimensionality compared to policy iteration methods and therefore are difficult to use in practical applications. The majority of model-free RL algorithms are however based on policy iteration that interleave policy evaluation and policy improvement steps [31]. Policy iteration methods may not have an explicitly parameterized value function. Policy iteration can be subdivided into on-policy and off-policy algorithms. On-policy RL algorithms generate experiences from the same policy which is under improvement, as opposed to off-policy algorithms that uses several different policies called "behavior policies" to generate experiences, while the target policy remains under improvement. Furthermore, RL algorithms can be classified as online or batch algorithms. In online RL algorithms, the policy and/or value function are updated at each new experience, while a whole batch of experiences are used to make an update in batch algorithms.

Policy and value iteration methods can be combined in different ways, resulting in a class of actor-critic methods [32]. An actor-critic method has both a parameterized policy (i.e. the actor) and a parameterized value function (i.e. the critic). In these methods, the gradient of the critic is used to update the actor.

C. Deterministic actor-critic policy gradient algorithm

A stochastic policy is often required to explore the complete state-space [19]. The policy gradient theorem [31] shows how the gradients of a policy can be written using approximate value functions. However, computing the stochastic policy gradient is cumbersome as the policy gradient iterates over both state- and action-spaces. Silver et al. [19] derives gradients of a deterministic policy which needs integration only over the state-space and is therefore faster for learning complex tasks

$$\nabla_{\theta} J(\pi_{\theta}) = \mathbb{E}_s[\nabla_{\theta} \pi_{\theta}(s) \nabla_a Q^{\pi}(s, a)|_{a=\pi_{\theta}(s)}]. \quad (2)$$

The deterministic policy may introduce a bias to the solution, and to compensate for that off-policy methods can be employed to ensure satisfactory exploration. However, an on-policy deterministic policy gradient algorithm may still be used if the experiment itself provides sufficient noise [19]. Silver et al. [19] also presents both on-policy and off-policy deterministic actor-critic algorithms.

Instead of online learning, which is known to be unstable with deep neural-networks, batch learning using experience reply are successfully employed in DQN [15] and DDPG algorithms [22]. The TD loss for the complete batch of experiences can be written using the mean-squared-error over k time-steps in the batch

$$\delta_L = \frac{1}{k} \sum_i (r_i + \gamma Q^w(s_{i+1}, a_{i+1}) - Q^w(s_i, a_i))^2, \quad (3)$$

where $Q^w(s, a) \approx Q^{\pi}(s, a)$, is a differential substitute of the true action-value function. Using the loss δ_L, the critic update ($\triangle w$) and actor update ($\triangle \theta$) for the deterministic actor-critic algorithm as presented in [19] are

$$\triangle w = \alpha_w \delta_L \nabla_w Q^w(s_t, a_t), \quad (4)$$

$$\triangle \theta = \alpha_{\theta} \nabla_{\theta} \mu_{\theta}(s_t) \nabla_a Q^w(s_t, a_t)|_{a=\mu_{\theta}(s)}, \quad (5)$$

where α_w, α_{θ} are learning rates and $\mu_{\theta}(s)$ is the deterministic policy to be learned.

III. Experiments

Previously proposed methods to automate bucket-filling did not result in industrial solutions because of difficulties in adapting these solution with changing conditions encountered in the construction and mining industry. We argue that an imitation learning based bucket-filling solution equipped with an RL algorithm is well suited to adapt and learn in changing operating conditions. To test this theory, the imitation learning based automatic bucket-filling solution trained with data from medium coarse gravel, see Fig. 5 (left), was adapted to a gravel-cobble pile (right) using the deterministic actor-critic algorithm with two different reward mechanisms.

It was observed that the baseline imitation model trained for medium coarse gravel successfully fills the bucket in a gravel-cobble pile with satisfactory performance measured by the productivity (tons/sec). The aim of reinforcement learning experiments is now to examine if the performance in the gravel-cobble pile can be improved further.

Fig. 5. Medium coarse gravel (left) and gravel-cobble material (right). Medium coarse gravel contains fine particles up to 64 mm, while gravel cobble material contains fine particles up to 200 mm.

A. Setup

We adapt the deterministic actor-critic algorithm proposed in [19] to the autonomous bucket-filling problem. The differences between computer simulation applications and our experiments with a 20-tonne machine force us to make practical choices regarding the experiment. Given the time and cost constraints, the experiment cannot be run for millions of time-steps. Thus, the experiment is run on-policy, without explicit exploration noise and in a batch mode (five bucket-fillings at a time), which limits the exploration-space but provides necessary constraints. The stochastic nature of the bucket-pile interaction introduces noise in the experiment. Thus, we decided to avoid the complexity of exploration noise.

The representation of the problem, i.e., the state- and action-space, is the same as for the imitation based learning method. The features $[F_L\ \theta_L\ \dot{\theta}_L\ F_T\ \theta_T\ \dot{\theta}_T\ \omega_{DA}]$ and their time-delayed values constitute the state. These features are scaled so that they are in similar ranges of $\sim [-2\ 2]$. The actions are the lift and tilt joystick values $[j_L\ j_T]$. Bidirectional joystick movements produce 0 to 5 volt outputs (where 0.7 to 2.3 V correspond to extension, 2.7 to 4.3 V correspond to retraction and the rest is deadzone), which is converted into velocity demand by the hydraulics. The useful part of the joystick action space for bucket-filling is the extension zone of 0.7 to 2.3 volts, which is transformed into a $[0\ 1]$ range, where one indicates maximum velocity demand. The state–action space is discrete due to the underlying 16-bit variables of the hardware interface, but it is not further discretized in our model.

The goal of a bucket-filling process is to fill the bucket with a specified amount (such as 75, 100 or 110%) in a minimum amount of time with minimum fuel consumption and without wheel-spin. However, there is a trade-off between complexity of the goal and complexity of the learning problem [18]. We assume that the bucket-filling problem is complex and motivates a simplification of the complex goal specified above.

A simpler goal for the RL-agent is to maximize the bucket-weight, which is known at the end of each bucket filling. As the agent expects a reward at every time-step of the bucket-filling process, the goal needs to be decomposed into a reward signal; this is known as the temporal credit assignment problem [33]. Reinforcement learning maximizes the cumulative sum of discounted rewards. In the bucket-filling context, the bucket-weight can be decomposed as a sum of productivity over the bucket-filling time. Thus, one option is to give productivity as the constant reward for all time-steps of the bucket-filling process and let the learning algorithm figure out the temporal relation between actions and reward. Since the experiment is implemented in batch mode, the productivity for each bucket-filling can be computed afterwards and used to generate the reward signal value.

In classical RL-setups, either the agents receives zero reward at each time-step and a high reward when the goal is achieved, or a small negative reward at each time-step and a high reward when the goal is achieved [31]. Taking inspiration from the general practices, we subtract a constant value of one from the productivity values to make these values negative. The negative rewards motivate the agent to finish the episodes (bucket-fillings) as quickly as possible as opposed to positive reward which might motivate the agent to never terminate the episodes. The results with constant productivity based reward are presented in sec. III-B.

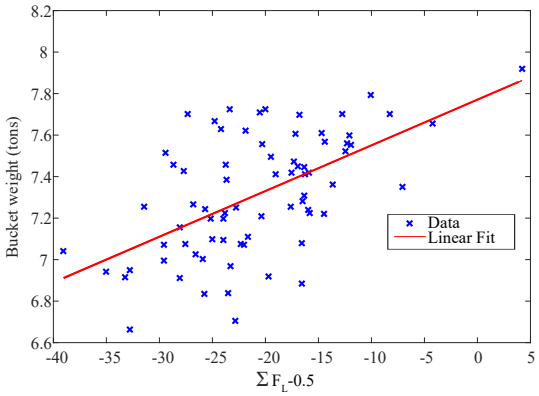

Fig. 6. Linear regression fit between higher lift impulse and final bucket-weight. The weak positive correlation ($R^2 = 0.35$) show that a high value of lift impulse is an indicator of high bucket weight. Thus, an RL reward based on the instantaneous value of the lift force was investigated.

In order to obtain a reasonable solution for a reinforcement learning problem, it is often needed to give intermediate reward in the reward signal; a process known as reward shaping [34]. Reward shaping is a way to incorporate domain knowledge into the setup so that the RL-agent is guided faster towards the goals [35]. To maximize bucket weights, both the domain knowledge and intuitions points towards increasing the force applied to the pile. Thus, we investigate the relation between the lift impulse and the final bucket weight for an old dataset generated by an expert operator, see Fig. 6. It appears that the impulse generated by high values of the lift force shows a weak positive correlation ($R^2 = 0.35$) with the final bucket weight. Consequently, we formulated a second reward signal $r_t = F_L - 1$. Again, a constant is subtracted to make the rewards negative, thus punishing the model when long times are spent in the bucket-filling process. An example of a reward signal, $F_L - 1$, is shown in Fig. 7, and sec. III-C presents the corresponding experimental results.

Similar to the imitation learning setup, we use time-delayed neural networks to implement both the actor and the critic. The actor network, see Fig. 4, has 12 units in the middle layer with softmax activation that are connected to class-based joystick outputs. The output layer of the actor network has two neurons, one each for lift and tilt predictions. The critic network, see Fig. 8, has 12 neurons in the hidden layer and one neuron in the output layer. Both the hidden and output layer of the critic network implement a tangent hyperbolic ($tanh$) activation function.

The original DQN [15] and the DDPG [22] algorithms use random mini-batches of experiences (s_t, a_t, r_t, s_{t+1}) from experience replay to do online training of the actor and critic

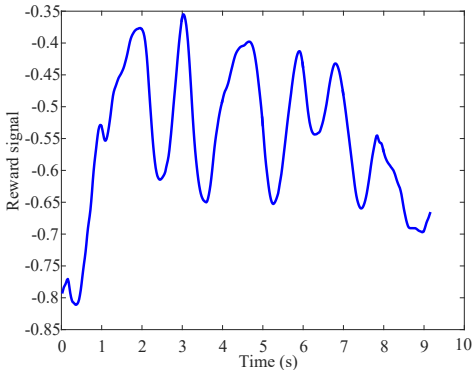

Fig. 7. An example of lift force based reward signal $F_L - 1$.

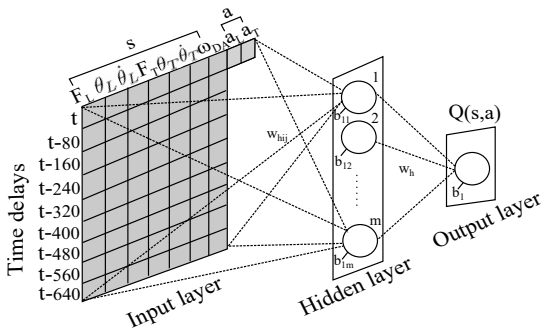

Fig. 8. The architecture of the critic network. The input layer, similar to the actor network, uses time-delayed value of features; the middle layer is a hidden layer with 12 neurons while the output layer has a single neuron, which predicts the action-value function.

networks. This is done to minimize the correlation between samples, which tends to slow down the learning process. Since we use a batch RL implementation, we randomize the complete batch experiences before the actor and critic update steps. The learning rates of both actor and critic network were 10^{-4} and the discount factor is kept constant at 0.9.

We use the tensorflow library [36] to implement and train our deterministic actor critic networks. The training of the actor and critic networks is done on a laptop. After each update the new parameters of the actor/critic are uploaded into a Simulink model which runs in a real-time PC connected to the wheel-loader ECUs. The base model in the real-time PC runs at 1 kHz, while the bucket-filling model runs at 50 Hz.

B. Productivity based reward signal

The experiment procedure with productivity reward signal in Fig. 9 depict that the actor is initialized the imitation learning model while the critic network is randomly initialized. In each cycle, data from $n = 5$ bucket-fillings is collected, arranged and reshuffled in an experience reply; the critic is updated using the TD loss from the Bellman equation and the actor is updated using critic's and its own gradients. We call each of this cycle as one DPG update.

The results following eight DPG updates for productivity reward signal in Fig. 10 show that the indented behavior is not obtained as the bucket-weights decrease and bucket-filling

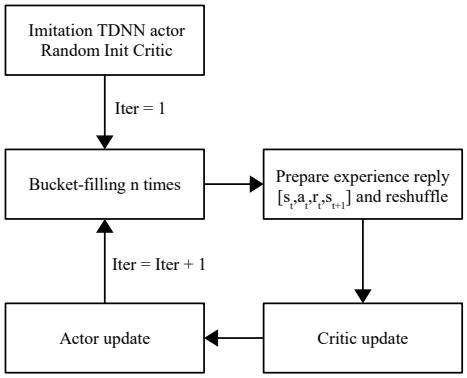

Fig. 9. Experiment procedure for the productivity reward signal. The actor is initialized with imitation learning model while the critic is randomly initialized.

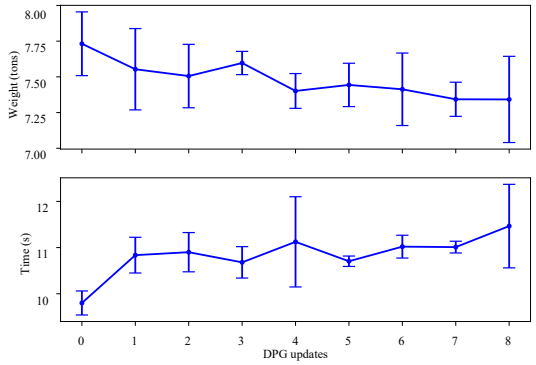

Fig. 10. Performance of deterministic actor-critic algorithm with productivity based reward signal. In first eight DPG iterations, the average weight of five test trails has decreased while bucket-filling time has increased which implies that the average productivity has decreased.

time increase during the limited duration of the experiment. We reason that (1) the productivity reward signal being constant in each bucket-filling has insufficient information for learning the critic and the actor, and (2) the critic network should initially be stabilized with more than five bucket-filling examples before the actor parameters are modified. Based on the preceding reasoning, the reward signal and the experiment procedure is modified.

C. Lift force based reward signal

In this experiment, we use a smooth reward signal $F_L - 1$ based on the reasoning that high values of lift force are positively correlated with final bucket weights and negative rewards motivate RL-agents to finish the episodes, implicitly incorporating an importance to shorter bucket-filling times. The experimental procedure using the lift force based reward signal described in Fig. 11 shows the initial (zero) iteration that was introduced to pretrain the critic network with $m = 13$ bucket-filling examples. In the zero iteration, the actor was not updated.

The results of this experiment in Fig. 12 show that the bucket-weights increase with DPG updates as compared to the imitation learning model (DPG = 0). The bucket-weights also

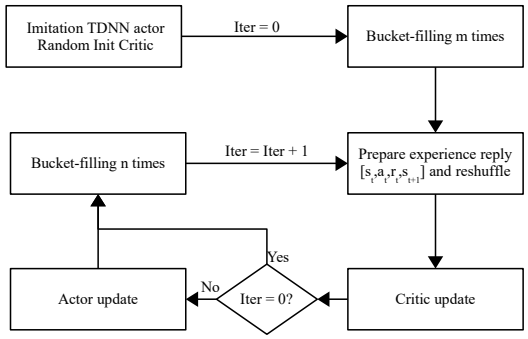

Fig. 11. Experiment procedure for the lift force reward signal. Zero iteration institutes pretraining of the critic network before the regular DPG updates.

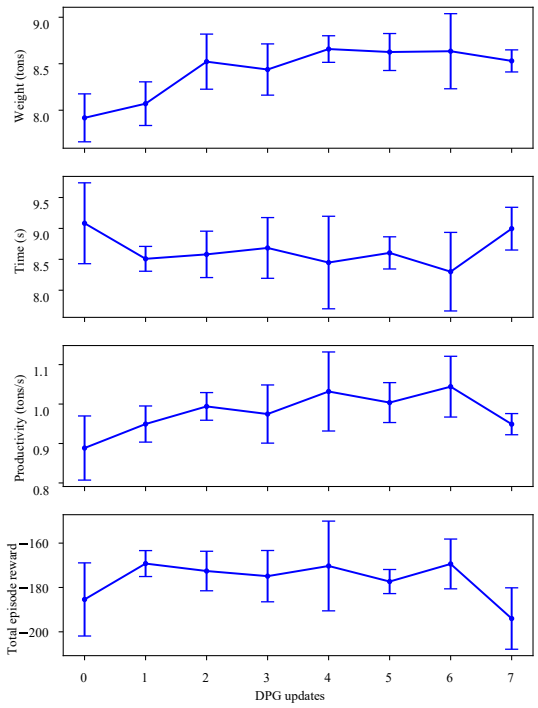

Fig. 12. Performance of deterministic actor-critic algorithm with lift-force based reward signal. The weights increase from an average of <8 tons ($n = 5$) with the imitation model to >8.5 tons, already with four DPG updates. The productivity plot follows the increments in bucket weights. A decrease of the bucket-filling time can be noticed for some DPG updates compared to the imitation based model, but this trend is not clear. The total episode reward plots mirror the bucket-filling time plot, as negative rewards implies that longer episode times result in lower cumulative rewards. Five bucket-filling operations were performed by the network in each DPG update.

exhibit saturation after four DPG updates indicating a local performance maxima. The bucket-filling times does not have a clear trend except for DPG updates = 1, 2, 5 and 6, which have significantly lower ($p<0.05$, Welch t-test [37]) bucket-filling times compared to the imitation learning model. Fig. 12 also shows the progress of productivity and total episode rewards with the DPG updates. It can be observed that the productivity increase follows the increase in bucket-weights while the total cumulative rewards reflect the bucket-filling time plot.

The lift force based reward function combined with pre-training of critic network has resulted in a fast learning implementation of deterministic actor-critic algorithm for the autonomous bucket-filling. The improvement in productivity is obtained in less than 40 bucket-filling examples, which corresponds to an hour of experiment time.

IV. Discussion

The ultimate goal of the bucket-filling algorithm is to consistently obtain a target bucket-weight in minimum time without wheel-spin. Here we consider the related problem to improve the imitation learning based bucket-filling model developed in former work [23] in a specific way in a novel pile environment using reinforcement learning. The type of improvement desired is defined by a reward signal. The reward signal plays an important role in reinforcement learning. The RL-agent receives rewards from the environment or a programmed reward function. Motivated by the observation that the bucket weights obtained with the imitation based model is low in novel pile environments, and the importance of the bucket weight for high productivity, we shape the reward signal to increase the bucket weight with minimum filling time.

We find that the use of a reward signal based on the lift force, rather than the final bucket weight, speeds up learning. This result is obtained by trial and error using domain knowledge. In principle, the lift force based reward signal, $L_F - 1$, aims to (1) increase the lift force and (2) decrease the bucket-filling time. The resulting increase of bucket-weights is an indirect effect of the trade off between a rewarding high lift force and early task completion.

Ng et al. [38] show that a potential based reward shaping signal $\gamma\Phi(s') - \Phi(s)$, where Φ is some function over states, do not change the fundamental learning problem and may increase the learning speed in many cases. The magnitude of the reward signal plays a role in training stability, as mentioned in [39] where a factor of 0.1 for rewards increase stability for the same experiments as those done in original DDPG paper [22]. As shown in Fig. 7, the values of the reward are not too big and training instability is avoided.

The discount factor, $0 \leq \gamma < 1$, balances between immediate and future rewards. If γ is zero, the learning is completely biased towards maximizing the immediate reward; however, typical values of γ are between 0.9 and 0.99. Here we set $\gamma = 0.9$, which apparently speeds up learning of the action-value function based on immediate rewards. This is desirable to decrease the number of experiments required to establish proof of concept of specific improvement of the imitation learning based bucket-filling model. Typically, the discount factor is kept constant; however François-Lavet et al. [40] shows that a gradually increasing discount factor, in combination with a gradually decreasing learning rate speeds up the learning of some Atari game playing models compared to the results in the original DQN paper [16].

The training of the critic with one batch of training examples likely plays a role for the results obtained with our lift force based reward signal. The experiments could for

TABLE I
TD LOSS PRIOR TO EACH CRITIC UPDATE ITERATION.

DPG update	0	1	2	3	4	5	6	7
TD-loss (δ_L)	0.617	0.121	0.116	0.120	0.119	0.120	0.125	0.140

practical reasons only be carried out with sufficiently fast learning of the critic network. The critic network aims to learn $Q(s,a) = \sum_{i=k}^{T} \gamma^{i-k} L_{Fi} - 1$, which is a discounted sum of lift force minus a constant. The lift force and some of its past values are included in the state, s, and thus is an input to the critic network. This makes it feasible to quickly learn a reasonable action-value function, $Q(s,a)$.

The value predicted by the critic in the training phase can be compared with true returns, as done in [22], to monitor the progress of critic network learning. Due to the use of a $tanh$ activation function, which limits the output range to $(-1,1)$ in our critic network, this direct comparison is unfortunately not possible. However, Table I show that the critic has converged already after the first update, as the TD-loss does not improve significantly after the first update. The TD loss corresponding to DPG–0 is the loss of a randomly initialized critic network. The decrease of the TD loss from DPG–0 to DPG–1 indicates that the critic network responds to training, while the nearly constant TD losses for subsequent DPG updates show that the critic network does not benefited from more training. The slight increase of the TD loss for DPG–7 is likely due to novel states appearing.

The last DPG update took longer time, as seen in Fig. 12, and the critic network has a higher TD-loss in the last DPG iteration, see Table I. The reason for this behavior could could be that the experimental machine is already operating at its maximum lift capacity, so that additional bucket weight increases lifting time and as a result produces novel data that is different from that in previous iterations, thus increasing the TD-loss. Alternatively, this can be explained by a changed and different shape of the pile, resulting in longer bucket-filling time and worse action-value estimates, thereby increasing the TD-loss.

In RL, inadequate exploration is associated with learning processes that get stuck in local minima, thus resulting in sub-optimal solutions. In the case of autonomous bucket-filling with real-world pile environments, we assume that the experiment itself contains enough variation. Thus, exploration noise is not introduced in the model considered here, which enables further simplification of the implementation. There are no light-weight simulation models for bucket-pile interactions suitable for closed-loop control. However, simulations based on the discrete element method (DEM) for bucket-filling exist [41], [42] and could be used in future work for learning and optimization of polices using exploration noise.

Unlike the related work on autonomous driving with RL [21], we do not use end-to-end camera based learning. In addition to vision, wheel-loader operators also interpret vestibular feedback and sound from the environment to perform efficient bucket-filling. Thus, for this type of learning it is not clear whether the addition of machine vision is motivated. Therefore, we consider a state representation including current and time-delayed values of standard machine signals, including forces on the lift and tilt pistons.

To our best knowledge, this is the first successful demonstration of a reinforcement learning algorithm on a full-size construction machine. The presented solution can potentially be scaled to a fleet of wheel-loaders with each machine initially configured with the same imitation model adapts to itself (geometry, engine, tires, bucket etc.) and its pile.

V. CONCLUSION AND FUTURE WORK

This paper demonstrates how reinforcement learning can be applied for adaptation of a wheel-loader automatic bucket-filling neural network pretrained with a few hours (100 examples) of imitation learning. This study contributes knowledge needed to automate the repetitive task of bucket-filling in earth-moving operation with front-end loaders. The results show that an automated bucket-filling model can be adapted to a different machine-pile environment using reinforcement learning, thereby avoiding the need for further imitation learning and retraining in the new environment.

The reinforcement learning architecture is based on a deterministic actor-critic algorithm [19]. Both the actor and the critic are implemented as time-delay neural networks, similar to the imitation learning based network. The experiment is run on-policy in batch mode with two different reward signals. The productivity based reward signal, which is constant for all time-steps of a bucket-filling operation, fails to improve the productivity during the experiment of about one hour. The proposed reward signal based on lift force varies continuously, and with pretraining of the critic network the bucket weights and productivity are observed to increase after only five updates of the deterministic policy gradient algorithm, requiring less than 40 bucket fillings.

Further work is required to verify that the bucket-filling model can be reliably adapted in different machine-pile environments using reinforcement learning. In particular, more work is needed to deal with more challenging materials such as clay and blasted rock. Furthermore, the bucket-filling reinforcement learning solution should be able to target a particular value of the bucket weight, while simultaneously learning to avoid wheel-spin. Reward signals that enable fast learning should be explored to address these complex goals. There is also a tradeoff between reliability and learning speed, which remains to be investigated in future work.

ACKNOWLEDGMENT

We thank Thomas Lind who is the wheel-loader operator that the initial neural network learned to imitate in this study.

REFERENCES

[1] A. Gustafson, "Dependability Assurance for Automatic Load Haul Dump Machines," Maintenance and Acoustics Division, Department of Civil, Environmental and Natural Resources Engineering Operation, Luleå University of Technology, Licentiate thesis, 2011.

[2] B. Frank, L. Skogh, and M. Alaküla, "On wheel loader fuel efficiency difference due to operator behaviour distribution," in *2nd Commercial Vehicle Technology Symposium*, Kaiserslautern, Germany, 2012, pp. 1–17.

[3] G. J. Maeda, "Learning and Reacting with Inaccurate Prediction: Applications to Autonomous Excavation," Ph.D. dissertation, School of Aerospace, Mechanical and Mechatronic Engineering, The University of Sydney, 2013.

[4] P. A. Mikhirev, "Theory of the working cycle of automated rock-loading machines of periodic action," *Soviet Mining Science*, vol. 19, no. 6, pp. 515–522, 1983.

[5] H. Almqvist, "Automatic bucket fill," Master's thesis, Department of Electrical Engineering, Linköping University, 2009.

[6] R. Filla, M. Obermayr, and B. Frank, "A study to compare trajectory generation algorithms for automatic bucket filling in wheel loaders," in *3rd Commercial Vehicle Technology Symposium*, 2014, pp. 357–374.

[7] J. A. Marshall, P. F. Murphy, and L. K. Daneshmend, "Toward autonomous excavation of fragmented rock: Full-scale experiments," *IEEE Transactions on Automation Science and Engineering*, vol. 5, no. 3, pp. 562–566, Jul. 2008.

[8] S. Sarata, H. Osumi, Y. Kawai, and F. Tomita, "Trajectory arrangement based on resistance force and shape of pile at scooping motion," in *IEEE Int. Conf. on Robotics and Automation*, vol. 4, 2004, pp. 3488–3493.

[9] W. Richardson-Little and C. Damaren, "Position accommodation and compliance control for robotic excavation," in *IEEE Conference on Control Applications*, 2005, pp. 1194–1199.

[10] A. A. Dobson, J. A. Marshall, and J. Larsson, "Admittance control for robotic loading: Design and experiments with a 1-tonne loader and a 14-tonne load-haul-dump machine," *Journal of Field Robotics*, vol. 34, no. 1, pp. 123–150, 2017.

[11] F. Y. Wang, "Agent-based control for fuzzy behavior programming in robotic excavation," *IEEE Transactions on Fuzzy Systems*, vol. 12, no. 4, pp. 540–548, 2004.

[12] P. J. A. Lever, "An automated digging control for a wheel loader," *Robotica*, vol. 19, no. 5, pp. 497–511, Aug. 2001.

[13] H. A. Fernando, J. A. Marshall, H. Almqvist, and J. Larsson, "Towards controlling bucket fill factor in robotic excavation by learning admittance control setpoints," in *11th International Conference on Field and Service Robotics*, 01 2018, pp. 35–48.

[14] J. Schmidhuber, "Deep Learning," *Scholarpedia*, vol. 10, no. 11, p. 32832, 2015, revision #184887.

[15] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[16] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[17] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel *et al.*, "Mastering chess and shogi by self-play with a general reinforcement learning algorithm," *arXiv preprint arXiv:1712.01815*, 2017.

[18] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement Learning in Robotics : A Survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[19] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *International Conference on Learning Representations*, 2014.

[20] R. Hafner and M. Riedmiller, "Reinforcement learning in feedback control," *Machine Learning*, vol. 84, no. 1, pp. 137–169, Jul 2011.

[21] A. Kendall, J. Hawke, D. Janz, P. Mazur, D. Reda, J.-M. Allen, V.-D. Lam, A. Bewley, and A. Shah, "Learning to drive in a day," *arXiv preprint arXiv:1807.00412*, 2018.

[22] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[23] 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.

[24] L. Torrey and J. Shavlik, "Transfer learning," in *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*. IGI Global, 2010, pp. 242–264.

[25] U. Andersson, "Automation and Traction Control of Articulated Vehicles," Ph.D. dissertation, Luleå University of Technology, 2013.

[26] S. Dadhich, U. Bodin, F. Sandin, and U. Andersson, "Machine learning approach to automatic bucket loading," in *24th Mediterranean Conference on Control and Automation*, June 2016, pp. 1260–1265.

[27] ——, "From tele-remote operation to semi-automated wheel-loader," *International Journal of Electrical and Electronic Engineering and Telecommunications*, vol. 4, no. 7, pp. 178–182, 2018.

[28] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang, "Phoneme recognition using time-delay neural networks," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 3, pp. 328–339, Mar 1989.

[29] S. Dadhich, F. Sandin, and U. Bodin, "Predicting bucket-filling control actions of a wheel-loader operator using a neural network ensemble," in *2018 International Joint Conference on Neural Networks (IJCNN)*, July 2018, pp. 1–6.

[30] L. Torgo and J. Gama, "Regression by classification," in *Advances in Artificial Intelligence*, D. L. Borges and C. A. A. Kaestner, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 51–60.

[31] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proceedings of the 12th International Conference on Neural Information Processing Systems*, ser. NIPS'99. Cambridge, MA, USA: MIT Press, 1999, pp. 1057–1063.

[32] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in *Advances in neural information processing systems*, 2000, pp. 1008–1014.

[33] R. S. Sutton, "Temporal credit assignment problem in reinforcement learning," Ph.D. dissertation, University of Massachusetts Amherst, 1984.

[34] A. D. Laud, "Theory and application of reward shaping in reinforcement learning," Ph.D. dissertation, University of Illinois at Urbana-Champaign, 2004.

[35] M. Grześ, "Reward shaping in episodic reinforcement learning," in *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, ser. AAMAS '17. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2017, pp. 565–573.

[36] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: A system for large-scale machine learning," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, 2016, pp. 265–283.

[37] B. L. Welch, "The generalization of student's problem when several different population variances are involved," *Biometrika*, vol. 34, no. 1-2, pp. 28–35, 1947.

[38] A. Y. Ng, D. Harada, and S. J. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *Proceedings of the Sixteenth International Conference on Machine Learning*, ser. ICML '99. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999, pp. 278–287.

[39] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," in *International Conference on Machine Learning*, 2016, pp. 1329–1338.

[40] V. François-Lavet, R. Fonteneau, and D. Ernst, "How to discount deep reinforcement learning: Towards new dynamic strategies," in *NIPS 2015 Workshop on Deep Reinforcement Learning*, 2015.

[41] R. Filla and B. Frank, "Towards finding the optimal bucket filling strategy through simulation," in *15th Scandinavian International Conference on Fluid Power 2017*. Linköping University, 2017.

[42] D. M. Lindmark and M. Servin, "Computational exploration of robotic rock loading," *Robotics and Autonomous Systems*, vol. 106, pp. 117 – 129, 2018.