Data-driven algorithms for predicting energy-efficient operating points in agricultural soil tillage

Benjamin Kazenwadel, Simon Becker, Lukas Michiels, Marcus Geimer
Institute of Mobile Machines (Mobima)
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
Contact: benjamin.kazenwadel@kit.edu

Abstract

Sustainability is especially important in power-intensive tasks in the agricultural industry, and therefore an aspect with optimization potential. During adjustments of the operating speed, fuel consumption can vary. The prediction of the most efficient operating point for the tractor-implement combinations is challenging due to the complexity of the machinery and the varying interaction forces between soil and machine. Currently, human drivers are required to manually adjust the operating speed. This paper presents and compares two optimization algorithms for predicting the most energy-efficient operating speed based on real-time measurements of the system state. Field tests were conducted to evaluate the algorithms under varying soil types and operating conditions. The algorithms accurately determined advantageous operating points within the defined working quality boundaries, making them promising tools for increasing sustainability in the agricultural industry.

Index Terms

Efficiency, Data-driven Algorithms, Optimization, Operating Point Prediction

Introduction

Data-driven algorithms, in particular artificial neural networks, offer high potential in the agricultural sector, especially in image processing, such as plant detection, and plant growth and health prediction. Further applications include weather analysis and yield prediction. [1]
The control of the drives in agricultural machines is complex and therefore under continuous research [2]. This paper demonstrates that data-driven algorithms can also be used for the optimization of these control tasks. We demonstrate the method by optimizing the operating speed during soil tillage. The operating speed influences the efficiency of the tractor and cultivator combination. The most efficient operating point depends on the operating conditions, including the soil composition, slope, and machine settings. Finding the most efficient operating point can be formulated as an optimization problem with the fuel efficiency η as the target function. The fuel efficiency is defined with the operating speed v_{real}, the working width w, and the fuel rate B.

$$v_{opt} = argmax(\eta) \tag{1}$$

$$\eta = \frac{v_{real} \cdot w}{B} \tag{2}$$

Figure 1 visualizes the target function for varying operating depths of a cultivator on even and parallel rows with constant soil composition. With an increase in depth, not only the overall efficiency decreases, but the optimal operating point shifts to lower speed settings.
In our previous research publications, we presented two approaches to solve this problem. One approach used off-policy reinforcement learning [3] and the other one used traction and draft modeling combined with a neural network for fuel rate prediction [4]. Both algorithms were limited to a discrete action space

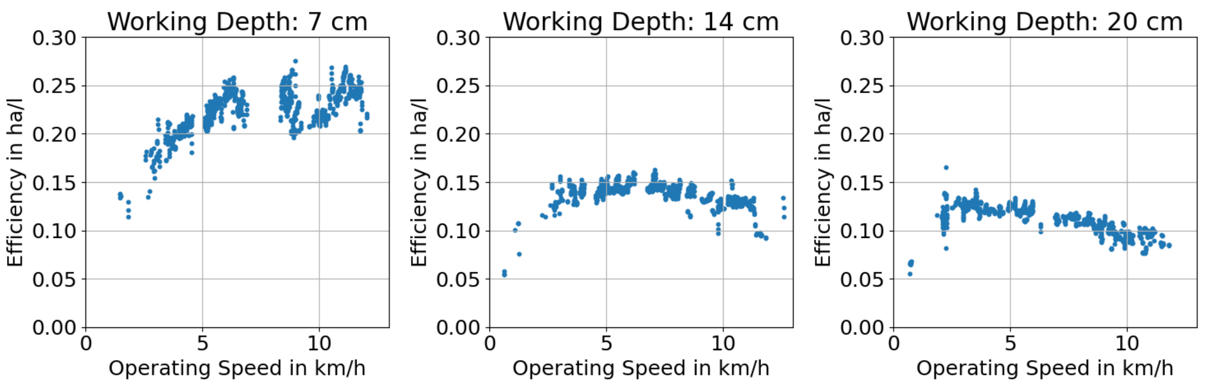

Fig. 1: Fuel Efficiency Example (Stationary Operating Points)

of accelerating, decelerating, and maintaining constant speed and could therefore not directly predict the optimal operating speed. This paper presents an offline reinforcement learning algorithm and an algorithm, which was specifically designed for agricultural tasks and uses an artificial neural network to predict the fuel consumption of the machine. Both algorithms were trained using a static dataset, which was created with a Fendt 724 tractor and a LEMKEN Karat 9 KUTA (4 m) cultivator driven by a human driver. The training dataset contains the signals of the internal CAN-Bus, which were merged by timestamp and filtered with a rolling mean filter. Furthermore, turning maneuvers were filtered out. The dataset contains approximately 82,000 individual machine states from 5.7 hours of fieldwork.

Algorithm 1: Conservative Q-Learning

Reinforcement learning algorithms can solve complex control tasks using data from the interactions of a machine with its environment. They can be divided into on-policy algorithms, where the policy π_k is trained with state transitions (state s_t, action a_t, next state s_{t+1} and reward r_t) from their current policy, whereas off-policy algorithms save several state transitions in a buffer and the data collection policy is not updated after each measured state transition. The updates are usually applied to the data collection policy after a few state transitions. Offline reinforcement learning algorithms, also known as batch learning algorithms, are adaptations of off-policy algorithms, developed to train the policy π_β with a static dataset and a different and unknown policy π_α during data collection. Direct interaction with the environment is therefore not required during the training process. Figure 2 visualizes these different approaches. [5]

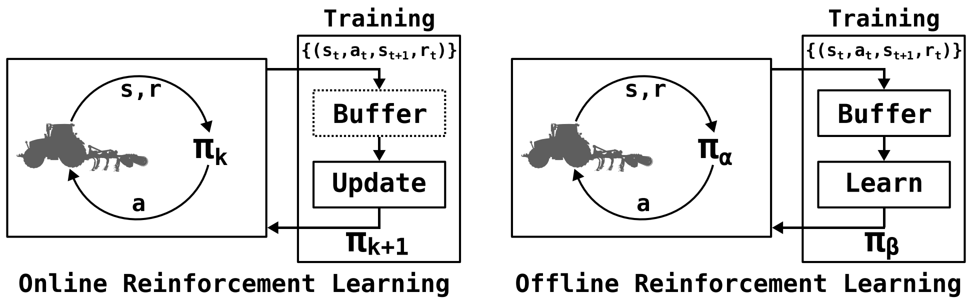

Fig. 2: Reinforcement Learning Algorithms, edited after [5]

For the training of online reinforcement learning algorithms, one option is to conduct extensive training in the field. This is unfeasible in agricultural processes as the costs of field experiments are high and appropriate operating quality cannot be guaranteed during the training process. An alternative is training in accurate simulation models, which require complex machine and environment models. Offline reinforcement learning algorithms solve these issues by being trained on a previously collected dataset, which is independent of the policy used during training data collection. Therefore, a dataset from a human operator can be used to train the included neural networks. Conservative Q-Learning (CQL) is one of these algorithms. The algorithm uses an additional loss term to counteract the effects of the prediction of out-of-distribution actions and the distributional shift between training and evaluation data [6]. The algorithm is an adaptation of the soft-actor critic algorithm (SAC) and is therefore applicable for continuous control problem [7]. Specific tests were implemented as a preselection strategy of promising trained control policies to reduce the number of required field experiments. Two parallel rows on an even field with uniformly distributed soil were driven manually, one at approximately 5 km/h and one at full throttle. Then the observation of these rows was played back and the proposed actions of the algorithm were compared, since in an ideal setting, the algorithm should propose the same optimal operating points on both rows. Furthermore, only fully trained policies were selected where the weights in the neural networks converged to static values. The choice of the reward function has a major impact on the generalization of the algorithm. The policies trained with the default target function (Equation (2)) failed to pass the preselection process. Therefore, we modified the reward function R of the CQL algorithm to optimize the specific power output by including the draft force F_D.

$$R = \frac{v_{real} \cdot F_D}{B} \tag{3}$$

The action vector $\vec{a}$ contains the theoretical speed v_{theo}. The state vector $\vec{s}$ contains the operating speed v_{real}, the theoretical speed v_{theo}, the draft force F_D, the slope δ, the acceleration a and the fuel rate B.

$$\vec{a} = (v_{theo})^T \tag{4}$$

$$\vec{s} = (v_{real}, v_{theo}, F_D, \delta, a, B)^T \tag{5}$$

The actions and observations were scaled to be in the range between 0 and 1. The algorithm was implemented in the open-source framework d3rlpy [8]. The architecture consists of two layers of 256 neurons for the actor-network and for the critic-network.

Algorithm 2: InterActNET

The second proposed algorithm is an analytic model for the interactions of the machine with its environment in combination with a neural network for efficiency prediction. The algorithm is visualized in Figure 3. This algorithm, called InterActNET, is built on the core principles of [4], surpassing the limita-

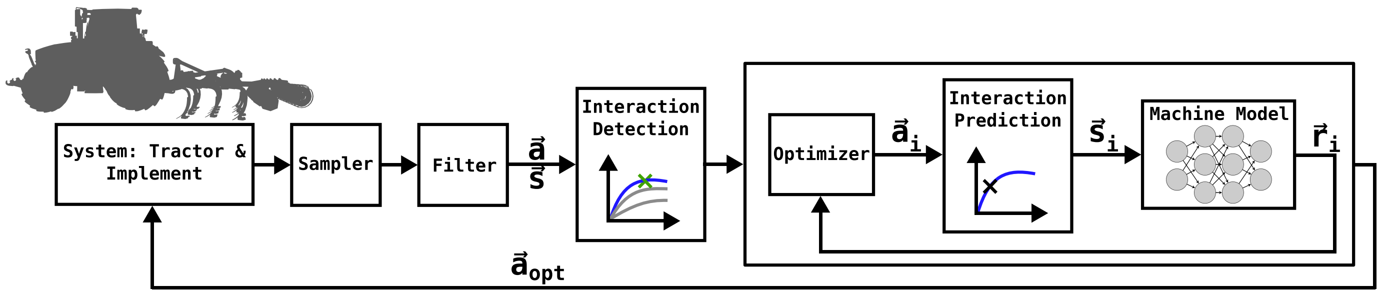

Fig. 3: InterActNET Optimization Architecture

tions of its predecessor by directly discretizing action spaces and increasing the speed and transferability of the algorithm. The algorithm assumes a quasi-stationary process where the variation of the optimal operating point as well as soil and slope conditions are negligible during the optimization interval of one

second. The algorithm identifies the current active interaction characteristics using action measurements $\vec{a}$ and state measurements $\vec{s}$ taken from the machine. Then, the optimizer suggests actions $\vec{a}_i$, of each of which the adjacent system states $\vec{s}_i$ are modeled using the identified interaction characteristics. The proposed actions and their respective states are entered into the machine model, which is represented by a neural network. This model returns a prediction of machine-internal variables $\vec{r}_i$. The resulting states and internal variables are compared in the optimizer using a target function (here: Equation (2)) and the best action $\vec{a}_{opt}$ is returned to the machine's internal controllers as a new target action.
The interaction model used in the interaction detection and prediction steps consists of a traction and a draft model, which are combined using the force equilibrium parallel to the surface under the neglection of the air resistance. In contrast to [4], the static force equilibrium parallel to the surface is adjusted by considering the weight forces of the tractor and implement separately and the inertia of the tractor and implement. The inertia is only used in the detection step and set to zero during the prediction step, since the goal is to predict stationary states. The weight forces are calculated using the slope angle δ and the tractor and implement weights, which were measured beforehand. The changing lifting forces of the rear links are neglected. The traction model assumes all-wheel drive, locked differentials, and that the overall traction characteristics of the tractor are similar to those of a single wheel. This assumption neglects the multi-pass effect and the effect of shifting weight forces on the axles. Furthermore, it assumes the same slip s for all wheels. The traction characteristics are modeled using the model by Brixius with the parameters for radial tires by Zoz and Grisso [9], [10]. In our algorithm, the included parameter c_T is identified in the interaction detection step, and the model can then be used for traction prediction.

$$F_T = F_{g,n} \cdot (1 - e^{-0.08 c_T}) \cdot (1 - e^{-7.0 s}) - \frac{1.2}{c_T} - \frac{0.5 s}{\sqrt{c_T}} \tag{6}$$

The interaction forces parallel to the ground between the implement and ground were modeled similarly to [4] using the proposed relationships by Harrigan and Rotz [11], [12]. The action and state vector are defined as

$$\vec{a} = (v_{theo})^T \tag{7}$$
$$\vec{s} = (v_{real}, F_D, a, \delta)^T \tag{8}$$

The machine model is a fully connected neural network with three layers of 128 neurons each, which is trained in a supervised fashion. The network was implemented in TensorFlow [13]. This reduces the modeling effort because machine internal characteristics, for example, the power-split transmission and the engine, do not have to be modeled manually or represented by a characteristic map. The input vector for the neural network combines a selection of the parameters from the action and the state vectors which were normalized between zero and one, whereas the output solely contains the fuel rate B:

$$\textit{input} = (v_{theo}, v_{real}, F_D, a)^T \tag{9}$$
$$\textit{output} = \vec{r} = (B)^T \tag{10}$$

The acceleration is set to zero during the prediction step due to the assumption of a quasi-stationary system state. The included neural network is trained by using a static dataset and skipping the interaction identification and prediction steps. The embedding of the acceleration in the training phase allows the use of non-stationary data points in the training phase. Therefore, the measured actions and states are directly used as input parameters, and the deviation between the predicted output vector and the measured output vector is used to train the neural network. The training process is described in Figure 4.
The optimizer evaluated the action space between 5 and 15 km/h which was discretized in 0.33 km/h steps. To avoid oscillation between accelerating and decelerating in the wide optimum of efficient states

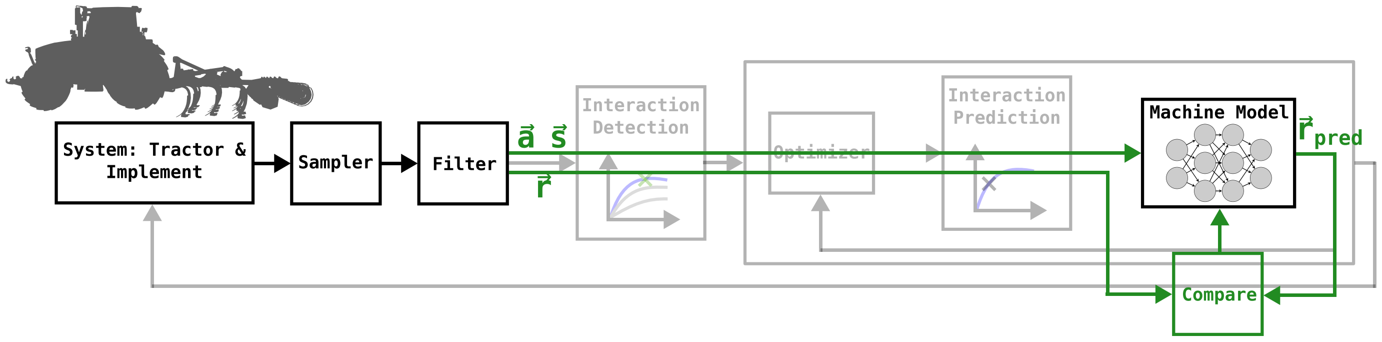

Fig. 4: InterActNET Machine Model Training

(Figure 1) we only allowed the adaptation of the target speed if the algorithm suggested an increase in efficiency of at least two percent compared to the previously selected action.

$$a_{opt} = \begin{cases} a_{opt,t-1}, & \eta_t(\vec{a}_{opt,t}) \leq 1.02 \cdot \eta_t(\vec{a}_{opt,t-1}) \\ a_{opt,t}, & \eta_t(\vec{a}_{opt,t}) > 1.02 \cdot \eta_t(\vec{a}_{opt,t-1}) \end{cases} \tag{11}$$

Evaluation

The evaluation was performed with the same Fendt 724 tractor and a LEMKEN Karat 10 (3 m) to demonstrate transferability. The test driver was a professional test driver who was tasked with driving parallel lines each for a reference line and then one line for each algorithm. Automatic steering and four-wheel drive were enabled. The algorithm runs were conducted fully automatically, and the driver did not interfere with the proposed settings other than setting a minimum and maximum speed limit. A total of 14 of these evaluation runs with varying depths and operating conditions were conducted, approximately one hectare for each algorithm.

Results

To achieve comparable results, the data points collected during the evaluation were averaged for each row. Since the evaluation conditions are not comparable between the evaluation runs, it is only possible to compare the relative differences with the human reference driver. Figure 5 visualizes the results.

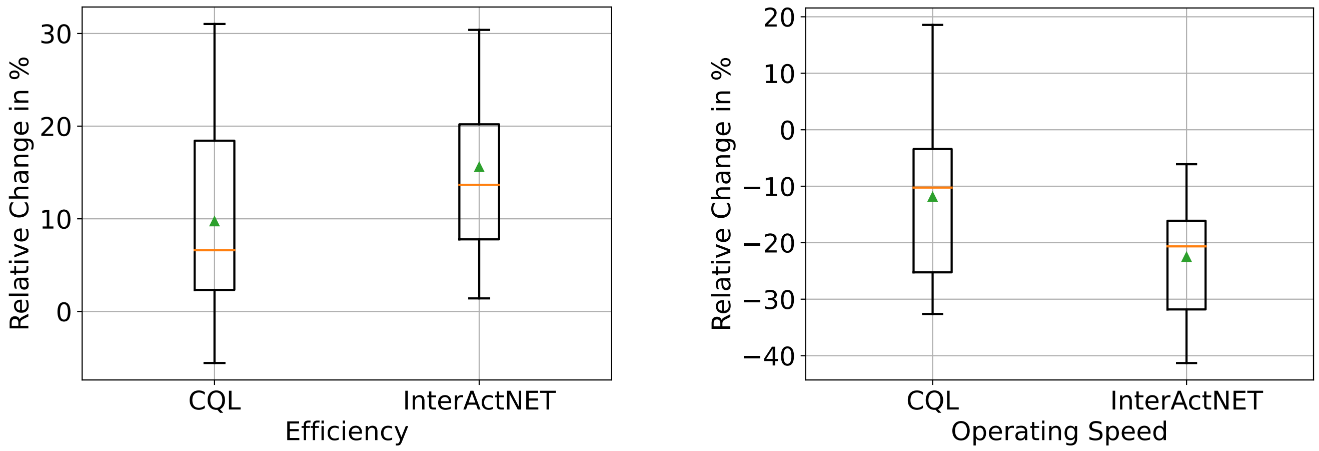

Fig. 5: Results

The CQL algorithm improved efficiency in eleven of the test rows with an average increase of 9.7%, which is represented by the green triangle. InterActNET was able to improve in all the test rows, with an average of 15.6%. In three of the test rows, the CQL algorithm suggested higher average speeds than the reference driver, and the respective rows did not fully coincide with the rows where CQL failed to optimize efficiency.
InterActNET suggested slower average operating speeds in all test rows. Both algorithms never selected actions outside the minimum and maximum speed boundaries. The test driver reported, that from a driver's perspective, the CQL algorithm suggests more pleasant speed changes. This can be explained by the fact that it uses state transitions and therefore acceleration characteristics during training, whereas InterActNET does not consider acceleration dynamics. Due to the overall suggestion of slower operating speeds, it can be argued that choosing the minimum operating speed regarding operating quality automatically maximizes fuel efficiency. However, this is dependent on the machine combination, settings, target function, and operating conditions (see Figure 1, Depth 7 cm).

Discussion and Outlook

Both proposed algorithms showed that they were able to predict more energy-efficient operating points. The offline reinforcement learning approach showed that purely data-based solutions can solve complex control optimization problems in agriculture. However, the solution is still limited to a specific target function and a specific task. This is not the case for InterActNET, where full states are predicted and implements and target functions can be exchanged. Possible improvements include updating the traction model to a wheel-specific model, taking soil compression due to the multi-pass effect, the front wheel advance, and the different wheel loads into account. Therefore, the lifting forces of the rear links of the tractor must be measured. Furthermore, the algorithm can be extended to include the PTO and the hydraulics in the optimization process.

Acknowledgements

We would like to thank AGCO for supporting this research and for conducting the field experiments. Furthermore, we would like to thank LEMKEN for providing the test cultivators.

References

[1] S. Kujawa and G. Niedbała, "Artificial Neural Networks in Agriculture," *Agriculture*, vol. 11, p. 497, June 2021. Number: 6 Publisher: Multidisciplinary Digital Publishing Institute.

[2] M. Geimer, *Mobile working machines*. Warrendale, Pennsylvania (USA): SAE International, 2020.

[3] S. Becker, K. Daiss, K. Daaboul, M. Geimer, and M. J. Zöllner, "Machine Learning for Process Automation of Mobile Machines in Field Applications," in *Land.Technik AgEng 2019 : Hannover, Nov. 8th + 9th 2019 : 77th International Conference on Agricultural Engineering*, p. 187, 2019. ISSN: 0083-5560.

[4] S. Becker, B. Kazenwadel, and M. Geimer, "Automation and Optimization of Working Speed and Depth in Agricultural Soil Tillage with a Model Predictive Control based on Machine Learning," in *LAND.TECHNIK 2022 The Forum for Agricultural Engineering Innovations*, (Online), pp. 55–64, VDI Verlag, Feb. 2022. ISSN: 0083-5560.

[5] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems," Nov. 2020. arXiv:2005.01643 [cs, stat].

[6] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative Q-Learning for Offline Reinforcement Learning," June 2020.

[7] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine, "Reinforcement Learning with Deep Energy-Based Policies," July 2017. arXiv:1702.08165 [cs].

[8] T. Seno and M. Imai, "d3rlpy: An Offline Deep Reinforcement Learning Library," Nov. 2021. arXiv:2111.03788 [cs].

[9] W. Brixius, "Traction prediction equations for bias ply tires," Tech. Rep. ASAE paper No. 87-1622, ASAE, St. Joseph MI, 1987.

[10] F. Zoz and R. Grisso, "Traction and Tractor Performance," *ASAE Distinguish. Ser.*, vol. 27, Mar. 2012.

[11] T. Harrigan and C. Rotz, "Draft Relationships for Tillage and Seeding Equipment," *Applied engineering in agriculture*, vol. 11, pp. 773–783, Nov. 1995.

[12] "ASAE D497.7 Agricultural Machinery Management Data," tech. rep., American Society of Agricultural and Biological Engineers (ASABE), 2011. Place: St. Joseph, MI Publisher: ASABE.

[13] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Y. Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems," 2015.