Model-free and Bayesian Ensembling Model-based Deep Reinforcement Learning for Particle Accelerator Control Demonstrated on the FERMI FEL

Simon Hirlaender*
IDA Lab, University of Salzburg, 5020 Salzburg, Austria

Niky Bruchon†
University of Trieste, 34127 Trieste, Italy
(Dated: December 17, 2020)

Reinforcement learning holds tremendous promise in accelerator controls. The primary goal of this paper is to show how this approach can be utilised on an operational level on accelerator physics problems. Despite the success of model-free reinforcement learning in several domains, sample-efficiency still is a bottle-neck, which might be encompassed by model-based methods. We compare well-suited purely model-based to model-free reinforcement learning applied to the intensity optimisation on the FERMI FEL system. We find that the model-based approach demonstrates higher representational power and sample-efficiency, while the asymptotic performance of the model-free method is slightly superior. The model-based algorithm is implemented in a DYNA-style using an uncertainty aware model, and the model-free algorithm is based on tailored deep Q-learning. In both cases, the algorithms were implemented in a way, which presents increased noise robustness as omnipresent in accelerator control problems. Code is released in `https://github.com/MathPhysSim/FERMI_RL_Paper`.

I. INTRODUCTION AND MOTIVATION

In particle accelerator operation, one main goal is to provide stable and reproducible performance. Achieving this demands the consideration of several control problems simultaneously [1, 2]. Especially if there is no way to model the physics a priori, one might use optimisation techniques as, e.g. derivative-free optimisers (DFOs) [3–8] or model-based optimisations as Gaussian processes [9, 10] to restore or maintain the performance.
Recently, the community has begun to explore a novel approach to solve control problems. Reinforcement learning (RL) [1, 11–14] unveils several advantages over optimisation methods:

- It covers a larger class of problems, as RL optimises a sequence of decisions.
- It memorises the problem and does not always begin from zero as a DFO.
- Already existing data can be used.
- Adaption to non-stationary systems is possible.
- The underlying structure of the problem might be deduced.

One demanding aspect using RL in real-world applications is the number of iterations needed to train a controller - the sample-efficiency, since RL methods are known to be data-hungry [15, 16]. A second critical aspect to be considered is the robustness of the training in a noisy data-limited environment.

In this paper, we present the study carried out to solve the maximisation-problem of the radiation intensity generated by a seeded Free-Electron Laser (FEL) on the Free Electron laser Radiation for Multidisciplinary Investigations (FERMI) at *Elettra Sincrotrone Trieste*. Figure 1 shows an aerial overview of the Elettra site and the FERMI free-electron laser buildings.
We successfully applied two different algorithm classes, a highly sample-efficient variant of continuous Q-learning (section III A 2) and a newly proposed model-based algorithm, which we call *AE-DYNA* (section III B), to the FERMI FEL problem.
The methods reveal different characteristics addressing the stated critical aspects. They are generally applicable to problems showing a sample-complexity as in common accelerator control problems.

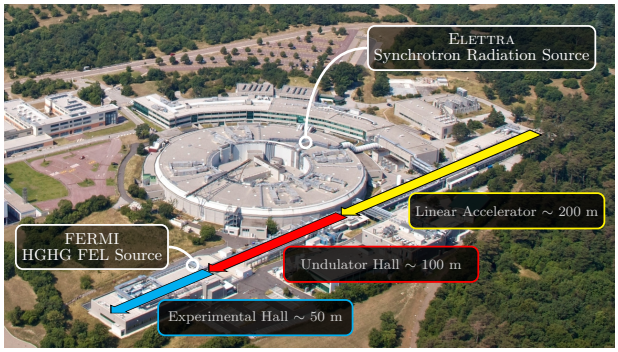

FIG. 1: The Elettra research centre hosting the FERMI free electron laser [17].

* simon.hirlaender@sbg.ac.at
† niky.bruchon@phd.units.it

A. An Overview of the Main Results

We demonstrate optimisation techniques that simultaneously adjust four parameters, the tilt and incline of two mirrors that control the trajectory and overlap of the seed laser in the FERMI FEL relative to the beam in the undulator. By making use of model ensembles and Bayesian neural networks our algorithms exhibit a high sample-efficiency (learning using < 1000 data sets), a low model-bias, and are extremely fast as they are able to maximize the FERMI FEL's output intensity to greater than 95% of the maximum within only 3-5 steps. Thus, we show the operational applicability of purely model-based reinforcement learning in an accelerator control problem.
Furthermore, our algorithms can perform real-time learning based on data and therefore, can be re-run and re-trained for time-varying systems.
The paper is organised as follows:

- Description of the problem set-up at FERMI
- Overview of RL
- Design decisions of the implementations used in these studies and theoretical concerns
- Results of the experiments
- Discussion, outlook and summary

II. THE SET-UP OF THE STUDIED PROBLEM

A. The Physical Set-up

In a seeded free-electron laser one of the most critical parameters is the temporal and transverse overlap of the electron and laser beam in the magnetic section called modulator undulator.
In the last years, various approaches have been studied to investigate their applicability in machine tuning.
A free-electron laser is a fourth-generation light source where the lasing medium consists of a very-high-speed electron moving freely through a magnetic structure. By using an external seeding source, the FERMI FEL has several advantages relative to standard FEL approaches in terms of increased stability in pulse and photon energy, reduced size, improved longitudinal coherence, more control on the pulse structure, and improved temporal synchronisation with external timing systems.
The external optical laser signal interacts with the relativistic electron beam in the undulator, introducing an energy modulation that aids in the FEL process. The modulation in energy is converted into a charge modulation in the dispersive section, and finally, the density modulated beams radiation is amplified in the section of the radiator.
The importance of ensuring the best possible overlap between the seed laser and the electron beam in the modulator is therefore evident.

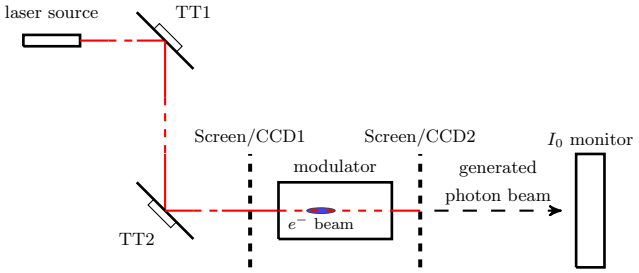

FIG. 2: A schematic view on the set-up of the FERMI FEL.

B. The Training Environment

A schematic overview of the set-up is provided in fig. 2. Two mirrors, TT1 and TT2, are used to control the trajectory of the laser by tilting and inclining, which gives a total of four degrees of freedom (DOF). In turn, the laser overlaps with the electron beam between the two removable screens, CCD1 and CCD2. Lastly, the monitor measures the intensity, I_0.
The final problem faced consists of optimising the seed laser trajectory to match the electron beam and consequently increase the intensity of the FEL radiation. The RL algorithm accesses the problem via an *openai gym* environment [18], which interfaces the RL algorithm with the control system.
The state $\mathbf{s}$ is a four dimensional vector holding the current voltage values applied to the piezo motors, where each component lies within the normalized interval [0, 1]. Two values correspond to the two DOF of each mirror. The action $\mathbf{a}$ is four dimensional, namely a delta step on the voltages at time-step t $\mathbf{s}_t$:

$$\mathbf{s}_{t+1} = \mathbf{s}_t + \mathbf{a}_t, \tag{1}$$

and is limited by an $|\mathbf{a}_t| \leq \mathbf{a}_{\max} = 1/12$[1]. The training is done in episodes with a uniformly[2] randomised initial state and a specific maximal length of taken steps afterwards - the horizon. For a reason discussed later, the horizon varies in the applications between either 10 or 500 steps. An early termination happens if a specific threshold is obtained, which lies in all cases at 95% of the maximal achievable intensity of the system. In our experiments, the maximal intensity was determined through manual expert optimisation.

[1] The value was determined experimentally to guarantee reproducibility.
[2] Sampled in the normalised state-space.

During the training phase exploration and the training of the function, approximator takes place. After a defined number of training episodes, verification of 50 or 100 episodes is performed. In the second phase, the RL only applies the learned policy.

III. DEEP REINFORCEMENT LEARNING

Most commonly, two main categories of deep RL algorithms are distinguished: model-free RL (MFRL) and model-based RL (MBRL). MFRL algorithms learn directly from the interaction with a system, while MBRL algorithms learn a dynamics model from these interactions first. Despite the success of MFRL in several domains as robotics and video games [19–23], their applications is mainly restricted to simulated environments due to their low sample-efficiency. However, putting constraints on the algorithms, as discussed in section III A, partly cures the sample inefficiency, turning them into practical tools for real-world problems.
On the other hand, MBRL showing a much better sample efficiency. The challenge for these methods is to learn an accurate model, which turns out to be rather hard and leads to the so-called model-bias problem [24]. In section III B we discuss attempts to capture the model uncertainty to alleviate this obstacle by making use of model ensembles and Bayesian neural networks resulting in algorithms, exhibiting a high sample-efficiency and a low model-bias.
In what follows, we denote the set of all states $\mathbf{s}$ by $\mathcal{S}$ and the set of all actions $\mathbf{a}$ by $\mathcal{A}$. Relative to state and action vectors, $(\mathbf{s}, \mathbf{a})$, we define a reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, a scalar $\gamma \in (0, 1]$ called discount factor and an initial state distribution d_0. $T(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$ characterises the probability to end in a state $\mathbf{s}_{t+1}$ if an action $\mathbf{a}_t$ is taken at state $\mathbf{s}_t$. The tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, T, d_0, r, \gamma)$ defines a Markov decision process.
Goal of reinforcement learning is to find a $\pi^* : \mathbf{s} \mapsto \mathbf{a}$, which is the solution of:

$$J(\pi^*) = \max J(\pi) = \mathbb{E}_{\tau \sim p_\pi} \left[\sum_{t=0}^{H} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t) \right], \tag{2}$$

where

$$p_\pi = d_0(\mathbf{s}_0) \prod_{t_0}^{H} \pi(\mathbf{a}_t, \mathbf{s}_t) T(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t) \tag{3}$$

is the distribution of all trajectories $\tau := (\mathbf{s}_0, \mathbf{a}_0, \mathbf{s}_1, \mathbf{a}_1, \ldots \mathbf{s}_H, \mathbf{a}_H)$ drawn by π with a horizon H.

A. Model-free Reinforcement Learning

In the modern field of RL one differentiates if the policy $\pi(\mathbf{a}) \approx \pi_\phi(\mathbf{s})$ or the state-action value function $Q(\mathbf{s}, \mathbf{a}) \approx Q_\theta(\mathbf{s}, \mathbf{a})$ is approximated using a high capacity function approximator, as e.g. a deep neural network.
In the first case π is optimized directly and one speaks about policy gradient methods [15, 20, 25–28]. To train these methods, a large number of direct interactions with the system is required, since they belong to *on-policy* algorithms. Hence, we rule them out for direct online training but make use of their characteristics in MBRL (section III C).
In the second case, we speak about Q-learning, which belongs to approximate dynamic programming. One advantage is the possible usage of use previously stored data as we use it in an *off-policy* fashion, which generally needs fewer data to be trained than *on-policy* methods. A combination of both is also used, known as actor-critic methods [21, 22, 29].
As already adduced, MFRL algorithms learn directly from interactions with the system. An exhaustive overview can be found, e.g. in [15, 30]. In the following, we discuss the normalised advantage function (NAF) algorithm [31] in some detail, as it has good characteristics for accelerator control problems. It is highly sample-efficiency, and its representational power is sufficient for most common cases[1, 32]. Modifications to increase the stability are subject of section III A 2.

1. Approximate dynamic programming

The state value function $V^\pi(\mathbf{s})$ tells us how good a state in terms of the expected return is following a specific π:

$$V^\pi(\mathbf{s}_t) := \mathbb{E}_{\tau \sim p_\pi(\tau|\mathbf{s}_t)} \left[\sum_{t=t'}^{H} \gamma^{(t'-t)} r(\mathbf{s}_t, \mathbf{a}_t) \right]. \tag{4}$$

The state-action-value function - short Q function - $Q^\pi(\mathbf{s}, \mathbf{a})$, which tells us how good an action $\mathbf{a}$ in a state $\mathbf{s}$ following π is, in terms of the expected return:

$$Q^\pi(\mathbf{s}_t, \mathbf{a}_t) := \mathbb{E}_{\tau \sim p_\pi(\tau|\mathbf{s}_t, \mathbf{a}_t)} \left[\sum_{t=t'}^{H} \gamma^{(t'-t)} r(\mathbf{s}_t, \mathbf{a}_t) \right], \tag{5}$$

is expressed as $Q^\pi_\theta(\mathbf{s}, \mathbf{a})$, where θ denotes the parameters of the function approximator, such as the weights of a neural network. By satisfying the Bellmann-optimality equation Q^π_θ can be trained towards the optimal $Q^*(\mathbf{s}, \mathbf{a})$ by minimizing the Bellman error[3]:

$$\min_\theta \left(\vec{Q}_\theta - \mathcal{B}^* \vec{Q}_\theta \right)^2. \tag{6}$$

π can be calculated via:

$$\pi_\theta(\mathbf{s}_t) = \delta(\mathbf{a}_t - \operatorname*{argmax}_{\mathbf{a}} Q_\theta(\mathbf{s}_t, \mathbf{a}_t)). \tag{7}$$

[3] $\vec{Q}_\theta$ denotes the Q-function Q_θ represented as a vector of length $|\mathcal{S}| \times |\mathcal{A}|$.

The Bellman operator $\mathcal{B}^*$ has a unique fixed point but is non-linear, due to the involved max - operator [15]:

$$\mathcal{B}^*\vec{Q}_\theta(\mathbf{s}_t, \mathbf{a}_t) := r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \max_{\mathbf{a}} \left(Q_\theta(\mathbf{s}_{t+1}, \mathbf{a}) - Q_\theta(\mathbf{s}_t, \mathbf{a}_t)\right). \quad (8)$$

The nature of this equation can cause overestimation and other complications, when using a function approximator and several attempts exist to overcome its difficulties [22, 31, 33–35].

2. Design decisions for MFRL

One way to avoid the sensitivity to the mentioned complications is to choose a simple analytical form with an explicitly calculable maximum of the Q-function. If a specific quadratic form of the Q-function is assumed [31]:

$$Q_\theta(\mathbf{s}, \mathbf{a}) = -\frac{1}{2}(\mathbf{a} - \mu_\theta(\mathbf{s}))P_\theta(\mathbf{s})(\mathbf{a} - \mu_\theta(\mathbf{s}))^T + V_\theta(\mathbf{s}). \quad (9)$$

$P_\theta(\mathbf{s})$ is a state-dependent, positive-definite square matrix, parametrised in a specific way [31]. One modification, which we made in our experiments is the application of a twin network (weights for network i denoted by θ^i). Only one network is used to obtain the policy, while the other is employed for the update rule to avoid over-estimation. It is motivated by double Q-learning [33, 36, 37]. The maximum is given analytically as $\max_{\mathbf{a}} Q(\mathbf{s}, \mathbf{a}) = V(\mathbf{s})$, hence from eq. (6) the loss $\mathcal{L}$ can be formulated as:

$$\mathcal{L}(\theta) = \left((r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \min_{1,2} V_{\theta^i_{\text{targ}}}(\mathbf{s}_{t+1}) - (1+\gamma)Q_\theta(\mathbf{s}_t, \mathbf{a}_t)\right)^2. \quad (10)$$

θ_{targ} are the weights of a target network, which is softly updated [21, 22, 31]. To further stabilize the network training a small clipped artificial noise is added to the actions called *smoothing* [37]. The effect is demonstrated in appendix A 1 on a classical example from control theory, the *inverted pendulum*. In fig. 14 its role to increase the noise robustness can be seen. Therefore *smoothing* was applied in all MFRL experiments.

Our implementation has high sample-efficiency and noise resistivity. We use it as the baseline for the FERMI-FEL control problem, as it yields good results. Additional changes to the original proposal [31] and a previous implementation used for accelerator control using to a prioritized replay buffer [32] are discussed in appendix A 1. We refer to our modified implementation as *NAF2* and the implementation close to the original, as in [31], as *NAF*.

B. Uncertainty Aware DYNA-style Reinforcement Learning

An excellent overview of the state of the art MBRL methods is provided in [38]. Suitable methods are [39, 40], based on Gaussian processes and Bayesian Neural Networks and back-propagation in the dynamics model or [41, 42], using highly efficient sample-based methods (e.g. the cross entropy method [43]). We focus on model-based data generation.

In MFRL the dynamics T of an environment is learned implicitly, while in model-based reinforcement learning MBRL T is learned explicitly and approximated as $\hat{T}$. In addition to the next state, the reward is included in the model [4]:

$$\hat{f}_\theta(\mathbf{s}_t, \mathbf{a}_t) := \{\hat{T}_\theta(\mathbf{s}_t, \mathbf{a}_t), r_\theta(\mathbf{s}_t, \mathbf{a}_t)\}. \quad (11)$$

Generally, DYNA style algorithms [44] denote algorithms, where an MFRL algorithm is trained on purely synthetic data from an approximate dynamics model $\hat{T}$ or a mixture of synthetic and real data. We use only synthetic data to reduce the interaction with the real environment to a minimum.

A schematic overview of our used method is shown in fig. 3. Initially, data is gathered by interaction with the real environment or read from a previous collection. An uncertainty aware model of the dynamics is trained, using 'anchored ensembling' [45] on the data (details provided in appendix A 2 and section III C 1), which allows taking the aleatoric (measurement errors) as well as the epistemic uncertainty (lack of data) into account. It was motivated by the fact that pure epistemic ensemble techniques as [46] were reported to be very sensitive to aleatoric noise [38]. Alternatives as [41, 42, 47] also take these uncertainties into consideration using probabilistic ensembles. Our implementation is simpler while showing excellent performance in considered experiments.

Subsequently, an MFRL algorithm is deployed to learn the controller on the dynamics model by only using the synthetic data and interoperating the model uncertainty, as explained in section III C 1.

After a defined number of training steps, called an epoch, the controller is validated on each model of the ensemble, individually. If there is no improvement in a number smaller than the total number of the models, the controller might be tested on the real environment for several episodes. In this way, 'over-fitting' on a wrong model is avoided, as discussed in [46]. The training is stopped if the controller produces satisfying results on the real environment. Otherwise, a batch of new data is collected, and the model is improved, and consecutively a new epoch starts.

---

[4] In some applications r can be deduced from the current state or the state change. In such a situation, the modelling can be simplified to model directly $\hat{T}$.

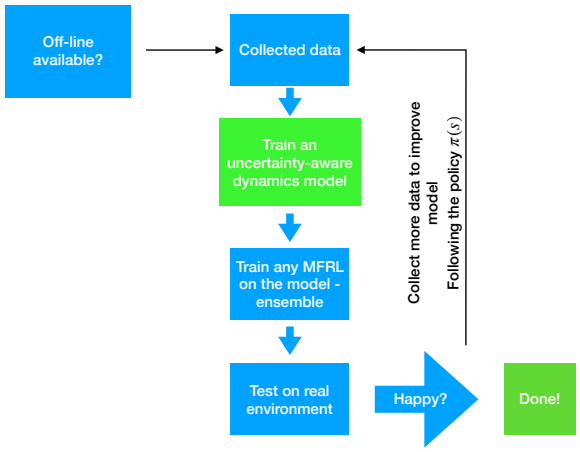

FIG. 3: A schematic overview of the *AE-DYNA* approach used in this paper.

C. Critical Design Decisions in MBRL

In the following, the most important aspects of a successful application of MBRL are discussed. To our knowledge uncertainty aware MBRL is applied for the first time with success on an accelerator problem and might be, despite its incredible potential, demanding. We made some algorithmic design decisions, which we think are beneficial for problems typically found in accelerator operation.

1. *The uncertainty aware dynamics model*

The 'anchored ensembling' method usually yields good results already with a small number of models. Empirical results showed that already three models were satisfactory to see definite improvements over a single network approach (see fig. 16). The main goal is not to determine the exact posterior probability of the model, but, as already mentioned, providing the uncertainty to not over-train the controller at areas without sufficient data. A small densely connected two-layer network with around 15-25 nodes each and *tanh* activation functions were used. The last layer was linear, and the inputs $\{\mathbf{s}, \mathbf{a}\}$ were normalised to the interval [-1,1].
The prior is controlled by the initialisation of the weights of the network. It defines how large the variation of the approximated function is expected and they were randomised uniformly in an interval $[-\Delta, \Delta]$, where $|\Delta| \leq 0.1$. The weights of the last layer were normalised by the number of its nodes. For the moment, only homoskedastic Gaussian errors are considered with a standard deviation σ_ϵ. This is respected in a regularisation term (details see appendix A 2) and significantly improves the training in the presence of noise, as shown in fig. 15.

Several methods of network training were implemented. *Early stopping* [48], with a learning rate of 10^{-3} to 10^{-4} was employed with a waiting time of about 20 steps and a validation ratio of 20%. The number of training steps is increasing with more data in close-by areas. It is followed by a shrinking of the uncertainty at regions where no data is available, leading to better training performance. A fixed maximal loss threshold was also beneficially tested. In the experiments, a combination of both was taken.

2. *The controller algorithm*

To decide which MFRL algorithm to use, two already mentioned main algorithm classes are considered (section III A): *on-* and *off-policy* algorithms [15]. *On-policy* algorithm show a stabler convergence in general, while *off-policy* algorithms have the advantage that the data buffer can be filled with real data.
The *on-policy* algorithm trust region policy optimisation (TRPO) [28] provides the theoretical guarantee of a monotonic improvement of the policy. We use this algorithm in experiments labelled as *ME-TRPO*, which stands for model-ensemble TRPO as proposed in [46]. As reported in [46], using proximal policy optimisation [20], another prominent *on-policy* algorithm, reveals minor performance, which was confirmed in our tests.
An attractive *off-policy* algorithm is the soft-actor-critic (*SAC*) [37, 49]. *SAC* not only tries to maximise eq. (2) but also simultaneously is regularised to maximise the entropy in the action space [50]. In this way, exploration is encouraged to avoid getting stuck in a local optimum, and a good trade-off between exploration and exploitation is achieved. We refer to experiments using the *SAC* as *AE-DYNA-SAC*. For *AE-DYNA-SAC* the controller was reset after each re-training of the dynamics model, to profit from its exploration features, while for *ME-TRPO* the policy was improved continuously to exploit its monotony.

3. *Handling of the uncertainty*

One of the most crucial points is how model uncertainties are taken into account by the MFRL controller. Several different approaches were tried. One way is to take the mean of the models μ_m and the standard deviation σ_m and sample from $\mathcal{N}(\mu_m, \sigma_m)$. Another strategy is to randomly select a model to provide at each training step, which demonstrated the best performance the original implementation of the model ensemble *TRPO* (*ME-TRPO*) algorithm and was used in the experiments labelled *ME-TRPO* [46]. A pessimistic selection would only select the model resulting in the lowest predicted reward (as used, e.g. in [51]) and showed constant but too slow improvement for our online-training. Good results were also obtained by following a randomly selected single model each full episode in combination, which was

used in the *AE-DYNA-SAC*.

4. The data acquisition

The number of data-points in a new batch to improve the model has to be chosen carefully. It is influenced by the number of randomly collected initial data-points (and then by the used policy π). The initialisation phase has to be selected not to small to minimise risks getting trapped in a local minimum for too long. Afterwards, from our experience, at least one full episode per epoch should be taken. We decided to use a short horizon to diminish the impact of the compound error [47], the accumulation error following a wrong model, as well known in *DYNA*-style approaches. The maximal number of steps in our experimental runs per episode was ten. During the data acquisition on the real system, the latest learned policy π was taken with some small additional Gaussian noise to improve exploration.

IV. EXPERIMENTAL RESULTS FROM FERMI RL ONLINE TESTS

As discussed in the previous sections, several tests were performed on the FERMI FEL. The primary purpose was to try the newly implemented algorithms on a real system to evaluate their operational feasibility. Because of the tight schedule at FERMI, only a few shifts of several hours could be reserved to carry out these experiments. Table I shows a comparison of the discussed algorithms. Details are provided in the appendix. All algorithms are suitable for control tasks as common in accelerator operation. Their strength lies in the high sample-efficiency, but they differ in representational power and noise robustness. In section IV the *NAF2*, *ME-TRPO* and *AE-DYNA* are tested and compared on the real environment. The *NAF2* algorithm, the representative for highly sample efficient MFRL algorithms, is discussed first.

TABLE I: Overview of the algorithms.

Algorithm	Class	Noise robustness	Represent. power	Sample efficiency
NAF	MFRL	low	low	high
NAF2	MFRL	high	low	high
ME-TRPO	MBRL	low	high	high
AE-DYNA	MBRL	high	high	high

A. MFRL Tests

In total, four tests were carried out, two using a single network and two using the double network architecture. Figure 4 displays the results, averaged over the two tests

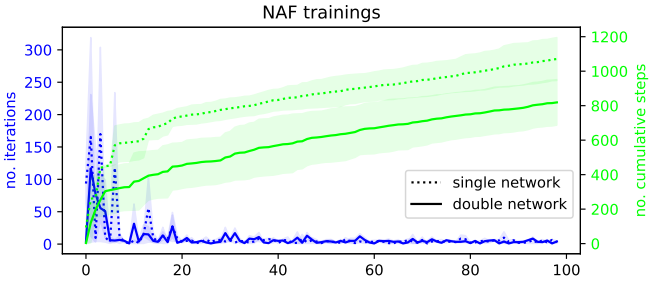

FIG. 4: The training of different variants of the *NAF2* algorithm on the FERMI FEL, averaged over two complete trainings (the standard-deviations are indicated by the shaded areas). The number of iterations (blue) shows the steps until the intensity is optimised, starting from a random initial position.

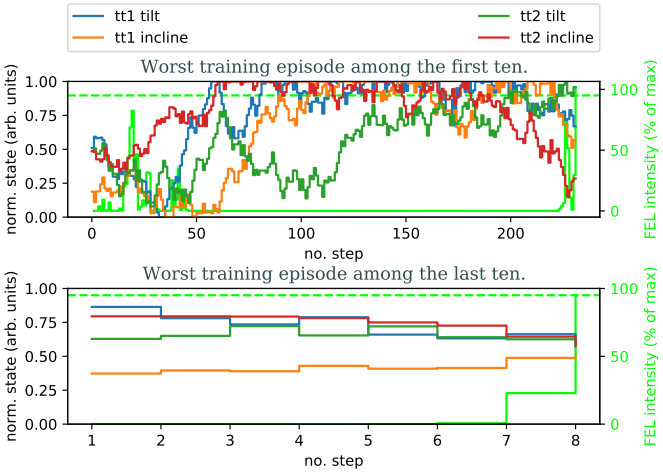

FIG. 5: The evolution of the states of the *NAF2* algorithm on the FERMI FEL using a double network during the training.

[5]. A training of 100 episodes was accomplished. The number of iterations per episode is plotted, including the cumulative number of steps. In all episodes, during the training, the 95% intensity reward threshold was surpassed, and therewith they were successfully finished. The evolution of the states is provided in figs. 5 and 6 for the single and double network. At the boundary of the domain, it takes time to learn, since many actions are mapped onto the same state[6]. The behaviour indicates that the double network converges faster and spends less time at the boundary.
In the verification of 100 episodes, fig. 7, both algo-

[5] Considering the training is stochastic with random initial conditions, by chance, the training of the double network variant started above the threshold in both cases.
[6] The states are clipped to $[0, 1]$.

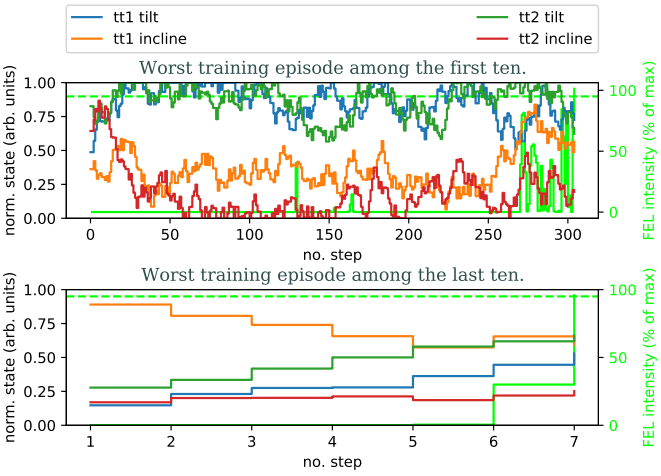

FIG. 6: The evolution of the states of the *NAF2* algorithm on the FERMI FEL using a single network during the training.

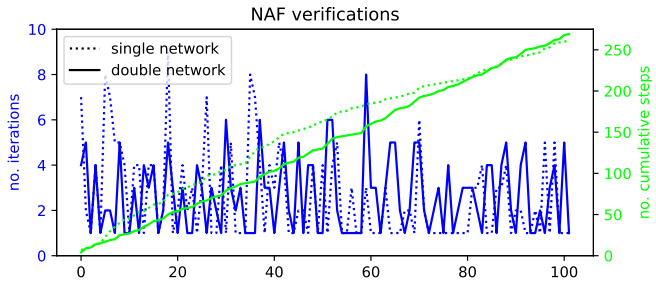

FIG. 7: The verification episodes of the variants of the trained model-free *NAF2* algorithm on the FERMI FEL. The number of iterations (blue) shows the steps until the intensity is optimised, starting from a random initial position.

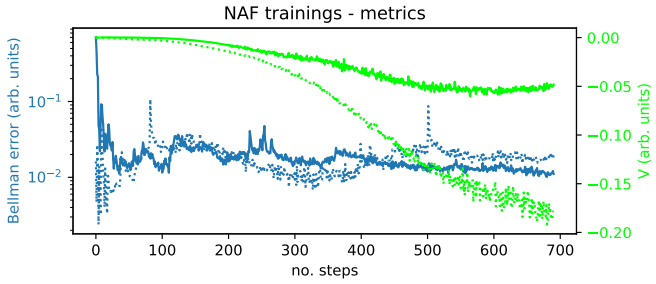

FIG. 8: The training metrics of the *AE-DYNA-SAC* on the FERMI FEL using a single network (dashed) and a double network (solid). The Bellman error (eq. (6)) and the state-value function (eq. (4)) are shown.

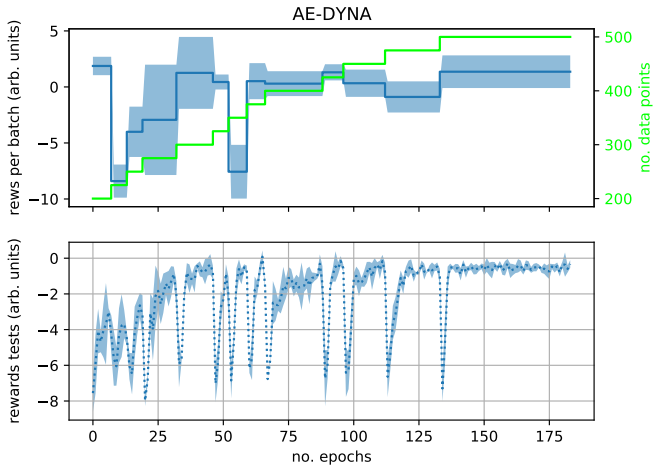

FIG. 9: The training observables of the *AE-DYNA-SAC* on the FERMI FEL. Detail are provided in the text.

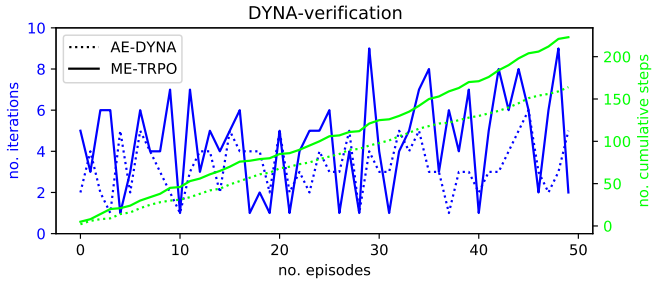

FIG. 10: The verification episodes of the trained model-based methods: *ME-TRPO* and *AE-DYNA-SAC* on the FERMI FEL. The number of iterations (blue) shows the steps until the intensity is optimised, starting from a random initial position.

rithms show similar performance, while the double network needed less training steps, and reveals a more stable overall performance. Additionally, the convergence metrics of the two algorithms is plotted in fig. 8 against the number of training steps. The blue curves show the Bellmann error eq. (6), which is comparable in both cases. The state-value function V (eq. (4)), which is a direct output of the neural net (eq. (9)), converges to a reasonable value for the double network within the shown 700 steps, whereas the single network seems to overestimate the value. In the single-network case, convergence is reached after around 1400 steps. In the plot, V was averaged over 200 randomly selected states.

B. MBRL Tests

The second test campaign was employing the *AE-DYNA* algorithm, as a representative for pure MBRL algorithms. As discussed, two variants were implemented:

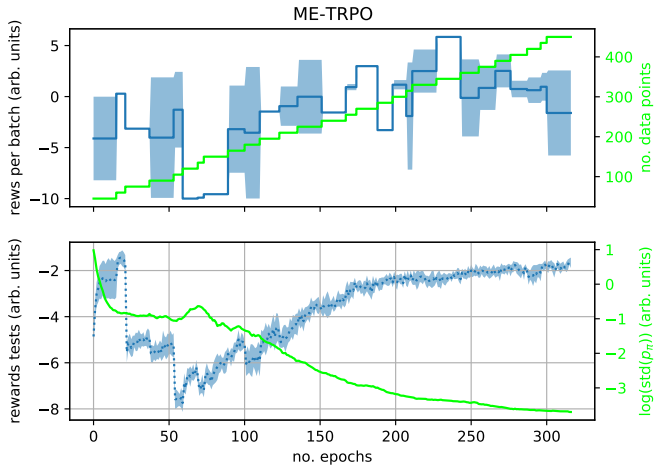

FIG. 11: The training observables of the *ME-TRPO* on the FERMI FEL. Detail are provided in the text.

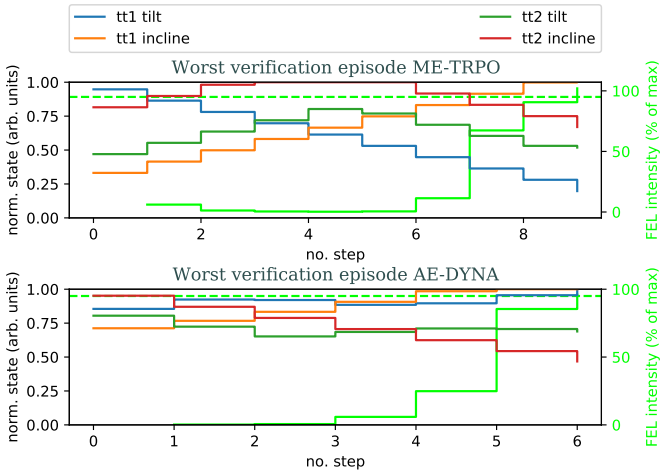

FIG. 12: The evolution of the states during the worst verification episodes of the trained *ME-TRPO* and the *AE-DYNA-SAC* on the FERMI FEL.

the *ME-TRPO* variant and the *AE-DYNA-SAC* variant. The algorithmic design details were discussed in section III C.

To exploit the convergence properties of the *TRPO*, the policy was never reset as can be seen in fig. 11. In the upper figure, the total reward per batch gathered from the real environment, and the number of data-points used to train the dynamics model as a function of the number of epochs is shown. In the lower plot, the average cumulative reward of ten episodes as achieved by the *TRPO* on the individual models of the ensemble independence of the epochs is drawn. During an epoch the *TRPO* is trained for 10000 steps on the synthetic data. The shaded area shows the corresponding standard deviation to indicate the uncertainty of the dynamics model. After initial performance drops caused by the insufficiently trained model, the algorithm continuously improves. As a measure of convergence of the *TRPO*, the logarithm of the standard deviation of p_π (eq. (3)) is visualised. The training was stopped after 450 steps collecting 25 steps each dynamics training.

As shown in fig. 10, all of the 50 verification episodes were successfully finished after a few steps. To verify the impact of the ensemble technique, a test with a single network using the same hyper-parameters has been applied, where no convergence within 500 data-points has been observed.

Secondly, the *AE-DYNA-SAC* was tested. As discussed in section III C, in this test, the controller was reset each time, when the model was re-trained and consequently, the performance drops each time as shown in fig. 9.

In contrast to the *ME-TRPO* training, the data batches consisted of 50 data-points with an initial random walk of 200 steps. The number of initial steps was chosen high enough because otherwise the convergence is slowed down strongly so that the training becomes unfeasible on a real machine (discussed in section III C). Each epoch consists of 2500 steps of controller training on the model. The training was stopped after the acquisition of 500 data-points. The verification was executed as in the first test. Again, the success of all 50 episodes is 100%. The number of needed iterations per episodes is less than for the ME-TRPO, which can be seen in fig. 10. The somewhat better performance might be a result of the higher number of data-points (50), but in general, this method exhibited better asymptotic performance than the *ME-TRPO* variant.

In both experiments, the training was using a small number of data-points while still successfully solving the problem. In fig. 12 the worst verification episodes of both experiments are plotted. The experiments, taken at different days, may influence the boundary conditions and hence the performance. However, longer training would increase the performance in both cases, especially for the *ME-TRPO*. Figure 12 shows that the tt2 incline and tilt parameters for the *ME-TRPO* move up and then down towards the optimal position, which would be improved by a better dynamics model.

V. DISCUSSION AND OUTLOOK

In this paper, the applicability of deep reinforcement learning on the optimisation of the FERMI FEL intensity was presented. Two different approaches were tested: model-free and model-based.

Both experiments yielded satisfactory results showing that a non-linear and noisy problem could be solved in a feasible number of training steps. The results of the verification are summarised in table II. In the model-free case, around 800-1000 and in the model-based 450-500 data-points were used. In all experiments, all verification episodes were finished successfully. *NAF2* performs bet-

ter in terms of episode length while the average reward in the *DYNA*-style algorithms is higher. The experiments were done on different days, hence under slightly different conditions.
Usage of the proposed methods in an operational way is attractive and could replace in the future the current optimisation method, which needs the destructive screen measurement. Hence an online retraining could be done and valuable time could be saved.
The MFRL methods were slightly better in the final performance, but also more samples have been collected during the training. The MBRL methods had to be stopped due to lack of available experimental time. Hence additional studies regarding a long time performance would be interesting.
One big issue in applying MBRL methods can be the computational time to train the controller on the model. In our tests, the data acquisition time was only a fraction of the time needed for the controller-training. In case, the used methods could be parallelised to reduce the computational time.

TABLE II: An overview over the verification performance of the different trained algorithms on the FERMI FEL including their standard deviation.

	Data points (counts)	Episode length (counts)	Cumulative reward (arb. units)	Final reward (arb. units)
AE-DYNA	500	3.28 ± 1.26	-1.44 ± 1.10	0.04 ± 0.06
ME-TRPO	450	4.46 ± 2.32	-1.95 ± 1.76	0.01 ± 0.04
NAF	1074	2.56 ± 1.96	-0.66 ± 1.22	0.00 ± 0.02
NAF2	824	2.64 ± 1.65	-0.57 ± 0.92	0.00 ± 0.03

VI. CONCLUSIONS

The presented reinforcement learning methods hold tremendous promise for automatizing set-ups typical in accelerators. One objective of this work was to provide some suggestions on how to make advanced deep reinforcement learning techniques work on a real set-up by adapting available methods. This was demonstrated on the FERMI FEL intensity optimization problem.
Regularly, control problems in accelerators have short horizons, which makes *DYNA*-style algorithms as the *ME-TRPO* or our *AE-DYNA* with their high representational power and excellent sample efficiency appealing choices.
Complementary, the *NAF2* algorithm presents a good alternative, revealing, as a model-free method, good asymptotic performance. As numerous cases in accelerator control can be captured assuming a quadratic dependence on the actions of the state-action value function, there is a broad spectrum of potential applications.
To provide the possibility for other laboratories to profit form the stated methods, the code was released in `https://github.com/MathPhysSim/FERMI_RL_Paper` and [52].

VII. ACKNOWLEDGEMENTS

We want to thank Giulio Gaio, who made these experiments possible and helped to set up and restore the FERMI FEL. Also, we are thankful for the valuable feedback from Alexander Scheinker on the draft of this publication. Finally, we want to thank the Land Salzburg for their financial support to create the IDA Lab, which made this work possible.

Appendix A: A Non-linear Standard Control Problem

To provide some transparency of these studies for other labs, we provide results on a famous classical standard control problem [53], the *inverted pendulum*.
It is a non-linear low dimensional unsolved continuous control problem. Unsolved means there is no threshold for the reward to terminate an episode. The episode length, the horizon, is set to 200 steps. In the following several tests were carried out on the *inverted pendulum* to demonstrate the improvements of the selected algorithms, mainly concerning the noise handling. It is of importance when dealing with measurements on real systems.
For statistical significance, all shown results were obtained using five different seeds. The average value and the standard deviation (shaded) are plotted. For this study, we assume that the problem is successfully solved if the cumulative reward surpasses a threshold of -200. A dashed green line indicates the threshold in the corresponding figures.

1. NAF2 Details

We compare the different *NAF* variants: *Clipping*, *No-clipping-smoothing*, *No-clipping-no-smoothing*, where *clipping* indicates the use of the double network, as introduced in section III A 2 and in all other cases a single network is used. The term *smoothing* indicates that a small clipped noise is added on the actions to stabilize the network training as:

$$\mathbf{a}(\mathbf{s}) = \text{clip}\left(\mu_{\theta_{\text{targ}}}(\mathbf{s}) + \text{clip}(\epsilon, -c, c), \mathbf{a}_{Low}, \mathbf{a}_{High}\right), \tag{A1}$$

where $\epsilon \sim \mathcal{N}(0, \sigma)$. $c > 0$ denotes the clipping coefficient and $\mathbf{a}_{Low}, \mathbf{a}_{High}$ the minimal and maximal possible action. This method was used already in [37] to improve the deterministic policy gradient [21]. The double network was used in [37, 50] and is done in the following way:

$$y(r_t, \mathbf{s}_{t+1}, d_t) = r_t + \gamma(1 - d_t) \min_{i=1,2} V_{\theta_{i,\text{targ}}}(\mathbf{s}_{t+1}), \tag{A2}$$

with d is 1 if the episode is finished and 0 otherwise. Then both are learned by regressing to this target (using the tuples from the data buffer $\mathcal{D}$):

$$L(\theta_i, \mathcal{D}) = \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1}, d_t)\sim\mathcal{D}} \left(Q_{\theta_i}(\mathbf{s}_t, \mathbf{a}_t) - y(r_t, \mathbf{s}_{t+1}, d_t) \right)^2, \quad i \in \{1, 2\} \quad \text{(A3)}$$

and the policy is obtained via $\max_{\mathbf{a}} Q_{\theta_1}(\mathbf{s}, \mathbf{a}) = \mu_{\theta_1}(\mathbf{s})$. The results are shown in fig. 13. One sees the cumulative reward per episode for a training of a total of 100 episodes. As mentioned, the curve labelled *clipping* corresponds to the double network, including *smoothing* and shows the best overall stability during the training yielding a high reward quickly. Also, the *smoothed* single network, labelled *No-clipping-smoothing*, shows good and comparable performance, except for the slightly decreased stability. The worst performance is achieved without smoothing and a single network (*No-clipping-no-smoothing*), nevertheless the result is competing with state of the art model-free methods as [54] as the benchmark in the *leaderboard* of openai gym [18].

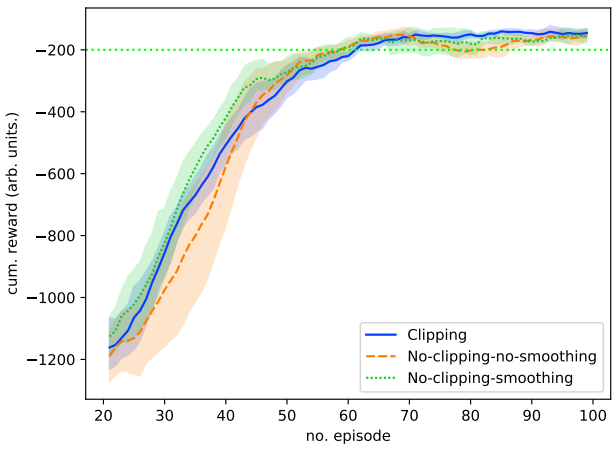

FIG. 13: Cumulative reward of different *NAF* implementations as discussed in the text on the *inverted pendulum* without noise.

2. The Impact of Noise

A test adding large artificial Gaussian noise $\epsilon \sim \mathcal{N}(0, \sigma_\epsilon)$ with $\sigma_\epsilon = 0.05$ in the normalized observation space on the states is presented in fig. 14. There the difference of the three methods becomes even more evident. The results are shown in fig. 14. After around 65 episodes the single network without *smoothing* (*No-clipping-no-smoothing*) decreases before reaching the final performance at around 95 episodes, while *smoothing* prevents this performance drop in the other cases.

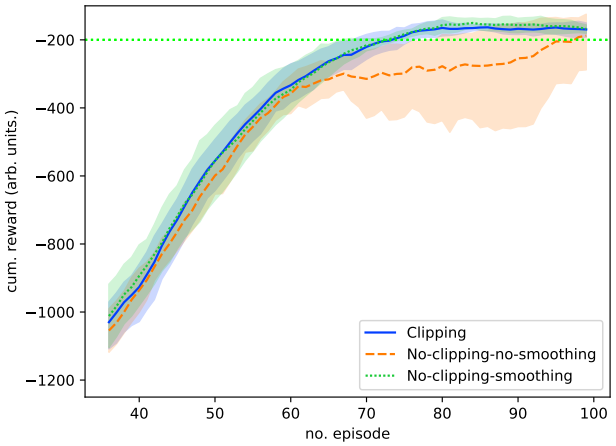

FIG. 14: Cumulative reward of different *NAF* implementations on the *inverted pendulum* with artificial noise as discussed in the text.

a. *Regression assuming homoskedastic Gaussian noise using 'anchored ensembling'*

The Bayesian community and the RL community is spending more and more attention to new approaches to Bayesian inference. In the 'anchored ensembling' technique a regularisation term is added to the loss function, which returns a point estimate of the Bayesian posterior the maximum a posteriori (MAP) parameter estimate. Using an ensemble of networks and adding noise on either the targets of the regularisation term a distribution of MAP solutions [45, 55–57] is obtained, which mimics true posterior.
Consider an artificial neural net containing parameters, θ, making predictions for the dynamics and the reward eq. (11), $\hat{f}_\theta$ (f denotes the true unknown function), with N data-points. If the prior is given by $P(\theta) = \mathcal{N}(\boldsymbol{\mu}_{prior}, \boldsymbol{\Sigma}_{prior})$, maximising the following returns MAP parameter estimates (details see [45]):

$$\theta_{MAP} = \text{argmax}_\theta \log(P_{\mathcal{D}}(\mathcal{D}|\theta)) - \frac{1}{2}\|\boldsymbol{\Sigma}_{prior}^{-1/2} \cdot (\theta - \boldsymbol{\mu}_{prior})\|_2^2 \quad \text{(A4)}$$

We replace $\boldsymbol{\mu}_{prior}$ with some random variable θ_{anc}. $\theta_{anc} \sim \mathcal{N}(\boldsymbol{\mu}_{prior}, \boldsymbol{\Sigma}_{prior})$, to make it practical and assuming homoskedastic Gaussian noise of variance σ_ϵ^2, we obtain a parametric form of the data likelihood. Taking M models, where each model is indexed by$j \in \{1...M\}$, the MAP estimates are found by minimising ($\mathcal{L}_j$ denotes the loss of the j^{th} model),

$$\mathcal{L}_j = \frac{1}{N}||f - \hat{f}_{\theta,j}||_2^2 + \frac{1}{N}||\boldsymbol{\Gamma}^{1/2} \cdot (\theta_j - \theta_{anc,j})||_2^2. \quad \text{(A5)}$$

$\mathbf{\Gamma}$ is diagonal regularisation matrix. The i^{th} diagonal element is the ratio of data noise of the target variable to prior variance for parameter θ_i:

$$\text{diag}(\mathbf{\Gamma})_i = \frac{\sigma_\epsilon^2}{\sigma_{prior_i}^2}. \tag{A6}$$

Using the anchors and $\sigma_\epsilon = 0.05$ in the dynamics model stabilizes the training of the *AE-DYNA* as illustrated in fig. 15. In the upper plot the mean cumulative reward on 10 test episodes on the real environment is shown during the training. It should indicate the result if the training is stopped at this training epoch. One cannot observe this quantity during a real training, unless one does costly performance measurements while training. Respecting the aleatoric (*Noise-on-aleatoric*) helps to reach the target much quicker exhibiting less variation compared to the standard use of an ensemble (*Noise-on-non-aleatoric*). An epoch consists of 3000 iterations of the *SAC*.
The lower plot of fig. 15 shows the batch rewards a measured during the data collection, which is observable during the training. Figure 16 shows the impact of the number of models onto the performance on the *inverted pendulum*. The maximum cumulative reward averaged over five different runs tested on the real environment during the training is visualized in dependence of the number of data-points. A number of three models (label *Three*) shows a good trade-off between performance and training time. A single network might not converge or too slowly (label *Single*) and a model of ten shown fast and stable performance in this case (label *Ten*).

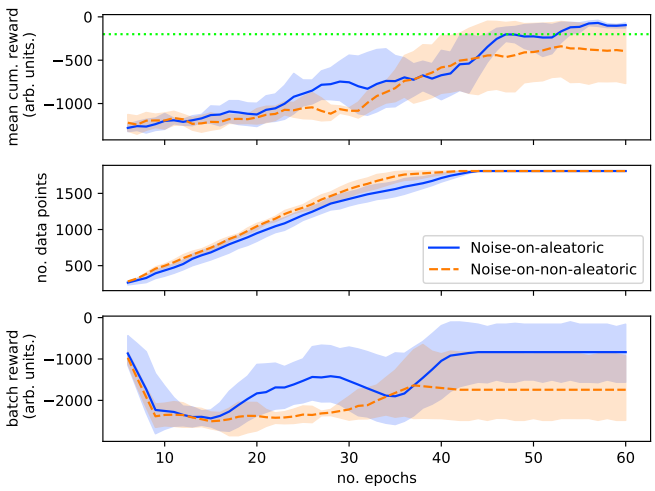

FIG. 15: Cumulative reward of *AE-DYNA-SAC* on the *inverted pendulum* with artificial noise using the 'anchor ensembling'.

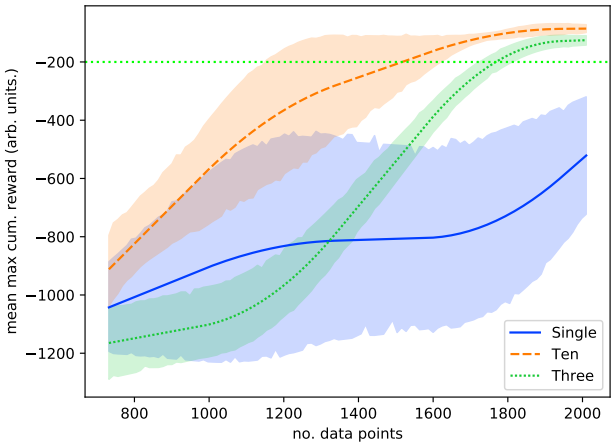

FIG. 16: Varying number of models in the ensemble of the *AE-DYNA-SAC* on the *inverted pendulum*.

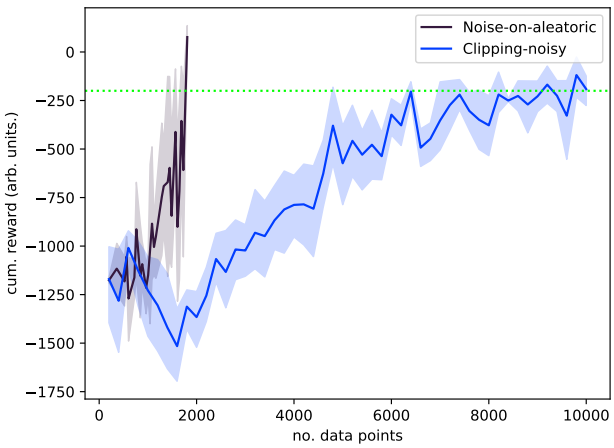

FIG. 17: The comparison of the *NAF2* and the *AE-DYNA-SAC* on the noisy *inverted pendulum*.

3. NAF versus AE-DYNA

Finally, fig. 17 demonstrates the sample-efficiency of the *AE-DYNA-SAC* and the *NAF* algorithm on the noisy *inverted pendulum*. *AE-DYNA-SAC* converges below 2000 data-points (without noise even below 800) and the *NAF2* starts to perform equally 10000 data-points surpassing the -200 reward threshold. One clearly sees the increased sample-efficiency on this problem using the *AE-DYNA* in contrast to the *NAF2*.

[1] V. Kain, S. Hirlander, B. Goddard, F. M. Velotti, G. Z. D. Porta, N. Bruchon, and G. Valentino, Sample-efficient reinforcement learning for CERN accelerator control, Physical Review Accelerators and Beams **23**, 10.1103/physrevaccelbeams.23.124801 (2020).

[2] A. Scheinker, A. Edelen, D. Bohler, C. Emma, and A. Lutman, Demonstration of model-independent control of the longitudinal phase space of electron beams in the linac-coherent light source with femtosecond resolution, Physical Review Letters **121**, 10.1103/physrevlett.121.044801 (2018).

[3] X. Huang, J. Corbett, J. Safranek, and J. Wu, An algorithm for online optimization of accelerators, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **726**, 77 (2013).

[4] N. Bruchon, G. Fenu, G. Gaio, M. Lonza, F. A. Pellegrino, and L. Saule, Free-electron laser spectrum evaluation and automatic optimization, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **871**, 20 (2017).

[5] A. Scheinker, S. Hirlaender, F. M. Velotti, S. Gessner, G. Z. D. Porta, V. Kain, B. Goddard, and R. Ramjiawan, Online multi-objective particle accelerator optimization of the AWAKE electron beam line for simultaneous emittance and orbit control, AIP Advances **10**, 055320 (2020).

[6] S. Hirlaender, M. Fraser, B. Goddard, V. Kain, J. Prieto, L. Stoel, M. Szakaly, and F. Velotti, Automatisation of the SPS ElectroStatic septa alignment, Proceedings of the 10th Int. Particle Accelerator Conf. **IPAC2019**, Australia (2019).

[7] C. Welsch, Numerical optimization of accelerators within opac, Proceedings of the 6th Int. Particle Accelerator Conf. **IPAC2015**, USA (2015).

[8] S. Albright, R. Alemany Fernandez, M. E. Angoletta, H. Bartosik, A. Beaumont, G. Bellodi, N. Biancacci, M. Bozzolan, M. Buzio, F. Di Lorenzo, A. Frassier, D. Gamba, S. Hirlander, A. Huschauer, V. Kain, G. Kotzian, D. Kuchler, A. Latina, T. Levens, E. Mahner, E. Manosperti, O. Marqversen, D. Moreno Garcia, D. Nicosia, M. O'Neil, E. Ozturk, A. Saa Hernandez, R. Scrivens, S. Jensen, G. A. Tranquille, C. Wetton, and M. Zampetakis, Review of LEIR operation in 2018, (2019).

[9] A. Hanuka, X. Huang, J. Shtalenkova, D. Kennedy, A. Edelen, V. R. Lalchand, D. Ratner, and J. Duris, Physics-informed gaussian process for online optimization of particle accelerators, (2020), 2009.03566.

[10] R. Roussel, A. Hanuka, and A. Edelen, Multi-objective bayesian optimization for accelerator tuning, (2020), 2010.09824.

[11] N. Bruchon, G. Fenu, G. Gaio, M. Lonza, F. H. O'Shea, F. A. Pellegrino, and E. Salvato, Basic reinforcement learning techniques to control the intensity of a seeded free-electron laser, Electronics **9**, 781 (2020).

[12] N. Bruchon, G. Fenu, G. Gaio, M. Lonza, F. A. Pellegrino, and E. Salvato, Toward the application of reinforcement learning to the intensity control of a seeded free-electron laser, in *2019 23rd International Conference on Mechatronics Technology (ICMT)* (IEEE, 2019).

[13] X. Pang, S. Thulasidasan, and L. Rybarcyk, Autonomous control of a particle accelerator using deep reinforcement learning, (2020), 2010.08141.

[14] J. S. John, C. Herwig, D. Kafkes, W. A. Pellico, G. N. Perdue, A. Quintero-Parra, B. A. Schupbach, K. Seiya, N. Tran, J. M. Duarte, Y. Huang, M. Schram, and R. Keller, Real-time artificial intelligence for accelerator control: a study at the fermilab booster, (2020), 2011.07371.

[15] R. Sutton and A. Barto, *Reinforcement Learning* (MIT Press Ltd, 2018).

[16] G. Dulac-Arnold, D. Mankowitz, and T. Hester, Challenges of real-world reinforcement learning, (2019), 1904.12901.

[17] N. Bruchon, *Feasibility Investigation on Several Reinforcement Learning Techniques to Improve the Performance of the FERMI Free-Electron Laser*, PhD thesis, Università degli Studi di Trieste (2020), (unpublished).

[18] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, Openai gym, (2016), 1606.01540.

[19] N. Heess, D. TB, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, S. M. A. Eslami, M. Riedmiller, and D. Silver, Emergence of locomotion behaviours in rich environments, (2017), 1707.02286.

[20] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, Proximal policy optimization algorithms, CoRR **abs/1707.06347** (2017), arXiv:1707.06347.

[21] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, Deterministic policy gradient algorithms, in *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML14 (JMLR.org, 2014) p. I–387–I–395.

[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, (2015), 1509.02971.

[23] OpenAI, M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba, Learning dexterous in-hand manipulation, (2018), 1808.00177.

[24] M. P. Deisenroth and C. E. Rasmussen, Pilco: A model-based and data-efficient approach to policy search, in *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11 (Omnipress, Madison, WI, USA, 2011) p. 465–472.

[25] R. J. Williams, Simple statistical gradient-following algorithms for connectionist reinforcement learning, Machine Learning **8**, 229 (1992).

[26] J. Baxter and P. L. Bartlett, Infinite-horizon policy-gradient estimation, 10.1613/jair.806 (2011), 1106.0665.

[27] S. Levine and V. Koltun, Guided policy search, in *Proceedings of the 30th International Conference on Machine Learning*, Proceedings of Machine Learning Research, Vol. 28, edited by S. Dasgupta and D. McAllester (PMLR, Atlanta, Georgia, USA, 2013) pp. 1–9.

[28] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, Trust region policy optimization, (2015), 1502.05477.

[29] C. Szepesvári, Algorithms for reinforcement learning, Synthesis Lectures on Artificial Intelligence and Machine Learning **4**, 1 (2010).

[30] S. Levine, A. Kumar, G. Tucker, and J. Fu, Offline reinforcement learning: tutorial, review, and perspectives on open problems, (2020), 2005.01643.

[31] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine, Continuous deep q-learning with model-based acceleration, (2016), 1603.00748.

[32] S. Hirlaender, Mathphyssim/per-naf: Initial release, 10.5281/zenodo.4271647 (2020).

[33] H. van Hasselt, A. Guez, and D. Silver, Deep reinforcement learning with double q-learning, (2015), 1509.06461.

[34] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, Playing atari with deep reinforcement learning, (2013), 1312.5602.

[35] Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, and N. de Freitas, Dueling network architectures for deep reinforcement learning, (2015), 1511.06581.

[36] H. Hasselt, Double q-learning, in *Advances in Neural Information Processing Systems*, Vol. 23, edited by J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta (Curran Associates, Inc., 2010) pp. 2613–2621.

[37] S. Fujimoto, H. van Hoof, and D. Meger, Addressing function approximation error in actor-critic methods (2018), arXiv:1802.09477 [cs.AI].

[38] T. Wang, X. Bao, I. Clavera, J. Hoang, Y. Wen, E. Langlois, S. Zhang, G. Zhang, P. Abbeel, and J. Ba, Benchmarking model-based reinforcement learning, (2019), 1907.02057v1.

[39] Y. Gal, R. McAllister, and C. E. Rasmussen, Improving PILCO with Bayesian neural network dynamics models, in *Data-Efficient Machine Learning workshop, International Conference on Machine Learning*, Vol. 4 (2016) p. 34.

[40] M. P. Deisenroth, D. Fox, and C. E. Rasmussen, Gaussian processes for data-efficient learning in robotics and control, IEEE Transactions on Pattern Analysis and Machine Intelligence **37**, 408 (2015).

[41] K. Chua, R. Calandra, R. McAllister, and S. Levine, Deep reinforcement learning in a handful of trials using probabilistic dynamics models, (2018), 1805.12114.

[42] T. Wang and J. Ba, Exploring model-based planning with policy networks, (2019), 1906.08649.

[43] P.-T. de Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein, A tutorial on the cross-entropy method, Annals of Operations Research **134**, 19 (2005).

[44] R. S. Sutton, Dyna, an integrated architecture for learning, planning, and reacting, ACM SIGART Bulletin **2**, 160 (1991).

[45] T. Pearce, F. Leibfried, A. Brintrup, M. Zaki, and A. Neely, Uncertainty in neural networks: approximately bayesian ensembling, (2018), 1810.05546.

[46] T. Kurutach, I. Clavera, Y. Duan, A. Tamar, and P. Abbeel, Model-ensemble trust-region policy optimization, (2018), 1802.10592.

[47] M. Janner, J. Fu, M. Zhang, and S. Levine, When to trust your model: Model-based policy optimization, (2019), 1906.08253.

[48] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT Press, 2016) `http://www.deeplearningbook.org`.

[49] A. Hill, A. Raffin, M. Ernestus, A. Gleave, A. Kanervisto, R. Traore, P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu, Stable baselines, `https://github.com/hill-a/stable-baselines` (2018).

[50] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine, Soft actor-critic algorithms and applications, (2018), 1812.05905.

[51] R. Kidambi, A. Rajeswaran, P. Netrapalli, and T. Joachims, Morel : model-based offline reinforcement learning, in *NeurIPS 2020* (ACM, 2020).

[52] S. Hirlaender and N. Bruchon, MathPhysSim/FERMI_RL_Paper: Initial release, 10.5281/ZENODO.4271580 (2020).

[53] K. Furuta, M. Yamakita, and S. Kobayashi, Swing up control of inverted pendulum, in *Proceedings IECON '91: 1991 International Conference on Industrial Electronics, Control and Instrumentation* (IEEE, 1991).

[54] G. Barth-Maron, M. W. Hoffman, D. Budden, W. Dabney, D. Horgan, D. TB, A. Muldal, N. Heess, and T. Lillicrap, Distributed distributional deterministic policy gradients, (2018), 1804.08617.

[55] Y. Gu and D. S. Oliver, An iterative ensemble kalman filter for multiphase fluid flow data assimilation, SPE Journal **12**, 438 (2007).

[56] Y. Chen and D. S. Oliver, Ensemble randomized maximum likelihood method as an iterative ensemble smoother, Mathematical Geosciences **44**, 1 (2011).

[57] J. M. Bardsley, MCMC-based image reconstruction with uncertainty quantification, SIAM Journal on Scientific Computing **34**, A1316 (2012).