Learning-based Optimisation of Particle Accelerators Under Partial Observability Without Real-World Training

Jan Kaiser [*1] **Oliver Stein** [*1] **Annika Eichler** [1]

Abstract

In recent work, it has been shown that reinforcement learning (RL) is capable of solving a variety of problems at sometimes super-human performance levels. But despite continued advances in the field, applying RL to complex real-world control and optimisation problems has proven difficult. In this contribution, we demonstrate how to successfully apply RL to the optimisation of a highly complex real-world machine – specifically a linear particle accelerator – in an only partially observable setting and without requiring training on the real machine. Our method outperforms conventional optimisation algorithms in both the achieved result and time taken as well as already achieving close to human-level performance. We expect that such automation of machine optimisation will push the limits of operability, increase machine availability and lead to a paradigm shift in how such machines are operated, ultimately facilitating advances in a variety of fields, such as science and medicine among many others.

1. Introduction

Reinforcement learning (RL) has been shown in various contributions as a potent method for solving complex tasks at super-human performance levels that were previously thought beyond the ability of computers (Silver et al., 2016; Badia et al., 2020). Furthermore, RL methods promise to find solutions faster for problems previously solved using classical optimisation algorithms by moving exploration from run time to design time. The application of RL to complex real-world tasks has however proven challenging (Irpan, 2018; Dulac-Arnold et al., 2019) for reasons such as the complexity of real-world tasks and sample efficiency.

Particle accelerators are an excellent example of a high-impact real-world application where RL can make a meaningful difference. Among the most advanced machines of our time, particle accelerators find use in many applications such as fundamental physics research, cancer treatment, the development of vaccines and drugs as well as the development and production of novel materials enabling for example for carbon-neutral transportation. These applications place strict requirements on the electron or photon beam delivered by the accelerator. Tuning accelerators to fulfil these requirements has historically been a challenging and difficult to automate task. As a result, accelerators continue to be tuned mostly manually by experienced human operators. Manual tuning is a lengthy process with hundreds of hours a year spent on tuning and therefore not available for productive operation at some facilities. Furthermore, the quality of the machine setup after tuning depends significantly on the operators' experience, limiting their reproducibility. Effective automation of accelerator tuning and optimisation has the potential to allow for faster tuning, make results easier to reproduce and possibly push the limits of accelerator operability. With the help of capable methods such as RL it is hoped that steps can be taken toward *autonomous accelerators* (Eichler et al., 2021) where operators no longer adjust actuators until a machine setup is achieved, but can instead define a desired setup that is then autonomously configured.

In this work, we demonstrate the application of RL to a real-world optimisation problem from a tuning task performed in regular operation of a linear particle accelerator. We show the trained RL agent's ability to solve the highly non-linear task under partial observability. Our solution achieves better results than classical optimisation algorithms and close to those achieved by experienced human operators. The RL agents manage to achieve these results in just a few steps, taking less time than human operators to perform the optimisation despite the RL agents being limited by slow hardware present on the particular real machine considered in this contribution. These excellent results are achieved on the real accelerator, using a 5-dimensional continuous action space, having trained fully in simulation and therefore without requiring any machine time for training. The transfer from simulation to machine is enabled by the inclusion of model errors into the simulation for a robust RL agent.

[*]Equal contribution [1]Deutsches Elektronen-Synchrotron DESY, Notkestr. 85, 22607 Hamburg, Germany. Correspondence to: Jan Kaiser <jan.kaiser@desy.de>.

Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

In the following, we summarise related work in Section 2. Then we give a brief introduction to RL in Section 3. In Section 4, we present our approach to applying RL to particle accelerator optimisation. Before concluding this work in Section 6, we present our results and compare them to classical optimisation in Section 5.

2. Related Work

The autonomous optimisation of particle accelerators during their operation is an active field of research (Eichler et al., 2021). Black-box optimisation algorithms in particular are applied to such problems both in research as well as in day-to-day operations. The software package *Ocelot Optimizer* (Tomin et al., 2016) for use in the control room streamlines the application of optimisation algorithms to accelerators and is regularly used in the operation of various facilities.

Because the hardware of particle accelerators may be slow to react, sample-efficiency is a key focus of research into the autonomous optimisation of accelerators. As a result, Bayesian Optimisation has found particular interest in the community and has been applied successfully to a variety of accelerator optimisation problems (McIntire et al., 2016; Hanuka et al., 2019; Kirschner et al., 2019; Shalloo et al., 2020; Duris et al., 2020). These optimisation methods are however often limited by either their speed or the complexity of the problems they can handle.

In an effort to improve the speed of classical optimisers on particle accelerators, other works explore the use of machine learning (ML) to aid the optimisation by fitting surrogate models to data from the real accelerator and performing optimisation on the latter (Edelen et al., 2020; Ivanov & Agapov, 2020).

The use of RL for autonomous particle accelerator optimisation is a young and fast growing research field. In (St. John et al., 2021), a Deep Q-Learning (DQN) agent is trained to act on a discretised continuous one-dimensional action space to regulate a gradient magnet power supply (GMPS) disturbed by environmental factors and improve its output stability. Using an artificial neural network (ANN) surrogate model for training and evaluation, the authors demonstrate that the RL performs better than an existing proportional–integral–derivative (PID) controller. In (Bruchon et al., 2020), two RL agents are trained on a real machine to attain and recover, respectively, high levels of self-amplified spontaneous emission (SASE) intensity at the FERMI free-electron laser at Elettra Sincrotrone. A high-dimensional highly observed and linear orbit correction problem with linear dynamics is solved with sample-efficient training of RL agents on a real accelerator in (Kain et al., 2020). In (Pang et al., 2020), continuous RL is used to optimise amplitude and phase settings of a drift tube linac (DTL) for optimal transmission of the beam in 3- and 5-dimensional continuous actions spaces, with 85 % and 21 % of the optimisations succeeding after up to 700 steps in simulation.

Outside of the accelerator community, there has been successful work on the *sim2real* transfer of simulation-trained ML systems exploring *domain randomisation* to train an object detector on randomly rendered images such that the real-world image appears as just another random image variation (Tobin et al., 2017). The concept of domain randomisation has also been applied to RL, where robot hands where trained to solve a Rubik's Cube from randomised image observations in simulation and then successfully applied in the real world (OpenAI et al., 2019). Most recently, a sim2real transfer in RL was successfully demonstrated on a more practical physical real-world application of controlling the plasma in a tokamak reactor (Degrave et al., 2022).

Compared to prior work on RL for accelerators, we consider a higher complexity problem both in terms of action space dimensionality and the underlying dynamics. We further successfully implement a sim2real transfer from a simulation-trained agent to the real accelerator with the help of domain randomisation, where prior works were either only evaluated in simulation or using a surrogate model, or used expensive beam time for training on the real accelerator. In addition, we are – to our knowledge – the first to compare the performance of an RL agent in accelerator optimisation directly to that of an expert human operator.

3. Reinforcement Learning

Reinforcement learning (RL) is a subfield of ML, where an *agent* is trained to iteratively solve a given task by maximising a *cumulative reward* R, the sum of immediate *rewards* r_t received in each step. In order to solve the task, the agent is provided an *observation* $\boldsymbol{o}_t$. The agent can use this observation to compute the next *action* $\boldsymbol{a}_t$ to take on the *environment* $\mathscr{E}$. In response to the action, the environment returns a new observation $\boldsymbol{o}_{t+1}$ and a reward r_{t+1}. The iteration, illustrated in Figure 1, is repeated indefinitely or until some terminal condition is met. Formally, we aim to solve a Markov decision process (MDP) by querying the agent's policy π_θ for the next action $\boldsymbol{a}_t = \pi_\theta(\boldsymbol{o}_t)$ and then performing this action on the environment that is currently in some internal state $\boldsymbol{s}_t$ as $\boldsymbol{o}_{t+1}, r_{t+1} = \mathscr{E}_{\boldsymbol{s}_t}(\boldsymbol{a}_t)$ to receive the next observation and reward. Many real-world problems are only partially observable, meaning that the internal state $\boldsymbol{s}_t$ is not fully reflected in the observation $\boldsymbol{o}_t$. In such a partially observable Markov decision process (POMDP) the Markov-property is not fulfilled and it may not be possible to predict the next observation $\boldsymbol{o}_{t+1}$ from a current observation-action pair $(\boldsymbol{o}_t, \boldsymbol{a}_t)$.

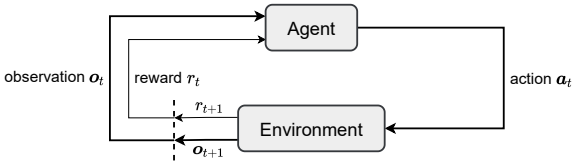


Figure 1. Flowchart of the RL loop. Based on the observation $\boldsymbol{o}_t$ of the environment, the agent chooses an action $\boldsymbol{a}_t$. As a result of the action, the environment transitions to a new internal state $\boldsymbol{s}_{t+1}$ and emits the next observation $\boldsymbol{o}_{t+1}$ and reward r_{t+1}.

The goal of training an RL agent is to find the optimal policy parameters θ^* that maximise the cumulative reward over all states of the environment. This is done by repeatedly letting the agent act on the environment and using the reward as a training signal to adapt the policy parameters via, for example, policy gradient or value-based RL training algorithms such as Proximal Policy Optimisation (PPO) (Schulman et al., 2017) or Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al., 2015).

4. Our Approach

Applying RL to optimisation problems on real-world machines requires the formulation of an optimisation problem as well as its translation to an RL problem. In the following we first introduce the problem setup and the corresponding optimisation problem in Section 4.1, followed by our RL-based solution of the problem. In developing this contribution, a number of challenges with the application to real-world machines in general have been addressed.

4.1. Experimental Area

In this contribution, we consider the optimisation of the transverse electron beam parameters in the Experimental Area (EA) section of the S-band radio frequency electron linear accelerator ARES (Panofski et al., 2021) at DESY in Hamburg, Germany. The EA is located downstream of the gun section and accelerating structures, where the electron beam is generated and accelerated to an energy of up to about 150 MeV, and ends with an experimental chamber that houses a variety of experiments. The EA is followed downstream by the second half of the accelerator. Both the experimental chamber as well the downstream section of the accelerator place stringent requirements on the beam properties at the end of the EA in order to enable state-of-the-art experiments.

The goal of this work is to optimise the transverse beam properties $\boldsymbol{b} = (\mu_x, \mu_y, \sigma_x, \sigma_y)$ on a diagnostic screen S toward the end of the EA, just upstream of the experimental chamber. Here, the properties μ_x and μ_y are the position of the beam, and σ_x and σ_y are the beam size in x- and

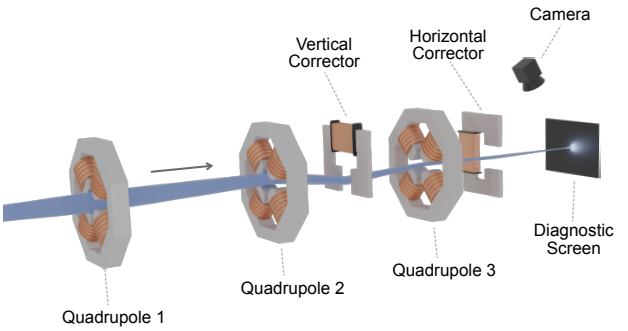


Figure 2. Illustration of the Experimental Area (EA) showing the components relevant for this contribution. The electron beam is indicated by the blue envelope. The beam enters the EA as indicated from the left.

y-direction within the transverse plane. They are calculated as the mean and standard deviation of the electrons in the beam, respectively. The electron beam is observed using a diagnostic screen made from a scintillating crystal material that emits light at an intensity corresponding to the number of impacting electrons. The image of the screen captured by a CCD camera allows the calculation of the position and size of the electron beam in the transverse plane of the screen S.

An arrangement of five magnets installed in the EA, three quadrupole magnets and two dipole magnets, may be used to manipulate the beam position and size. Quadrupole magnets act much like lenses in that they focus or defocus the beam. As a result of the quadrupole geometry, the effect is always reversed in the x- and y-planes, i.e. focusing in x means defocusing in y and vice versa. The magnitude of the focusing effect of a quadrupole is given by its *strength* k. The dipole magnets, so-called *correctors*, deflect within a plane according to their own orientation by some deflection angle α. In the EA there is a vertical and a horizontal corrector deflecting the beam in the vertical and horizontal plane, respectively. Figure 2 shows a simplified overview of the relevant components and their arrangement in the EA.

Depending on the experiment, different requirements are placed on the beam. We therefore consider in this contribution the problem of optimising the actuator settings $\boldsymbol{x} = (k_{Q_1}, k_{Q_2}, k_{Q_3}, \alpha_{C_v}, \alpha_{C_h})$ to adjust the transverse parameters of the beam on screen S to any set of beam parameters $\boldsymbol{b}'$ chosen by the operator as required for the present experiment. This problem can be formulated as an optimisation problem

$$x^* = \text{argmin}_{\boldsymbol{x}} D(\boldsymbol{b}, \boldsymbol{b}')$$

to find a vector of optimal actuator settings $\boldsymbol{x}$ that minimise the difference D between the observed beam parameters $\boldsymbol{b}$ and the desired beam parameters $\boldsymbol{b}'$.

To this end, we know that there exists a function $\boldsymbol{b} = F_{I,M}(\boldsymbol{x})$ that maps the actuator values $\boldsymbol{x}$ currently set on the magnets to beam parameters $\boldsymbol{b}$ measured on the diagnostic screen. F depends on two unknown stochastic variables, the beam entering the EA I and a set of potential errors in the constructed machine M, such as misalignments of the magnets as well as drifts. In the real world, neither the incoming beam I nor the machine errors M are known. Furthermore, a variety of other effects influence the beam in the real world, resulting in unknown deviations from the known function $F_{I,M}$. As a result $F_{I,M}$ is considered a stochastic black-box function.

4.2. Learning-based Optimisation

The described optimisation problem is reformulated as an RL problem in order to train agents to perform the optimisation. Four key components are required for an RL problem: action, observation, reward and environment.

The **action** defines how the agent interacts with the environment. For the optimisation task in the EA, the action may intuitively be defined as the magnet settings $\boldsymbol{a}_t = \boldsymbol{x}_t$. Our experiments have however shown that while this action definition works well in simulation, it stops working when transferring simulation-trained agents to the real world. This is because in the real world, effects, such as magnet hysteresis[1] and environmental factors, disturb the beam resulting from a particular set of magnet settings. The implication is that on the real accelerator, even if the environment were fully-observed, the same set of magnet settings may lead to different beams on the diagnostic screen at different times. We solve this by defining a continuous delta action $\boldsymbol{a}_t = \left(\Delta k_{Q_1}, \Delta k_{Q_2}, \Delta k_{Q_3}, \Delta\alpha_{C_v}, \Delta\alpha_{C_h}\right)$ as the amounts by which to change the actuator values $\boldsymbol{x}_t$ at time step t. At time step $t+1$, this results in actuator values $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \boldsymbol{a}_t$. This action definition allows the agent to dynamically compensate for unexplained deviations from the expected beam and is very similar to the way human operators interact with the magnets. Our experiments have shown that the latter definition enables a successful sim2real transfer.

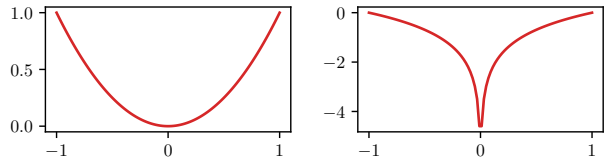
The **observation** is based on the internal state $\boldsymbol{s}_t$ of the environment. In the ARES EA, $\boldsymbol{s}_t$ can be mostly described by the current actuator values $\boldsymbol{x}_t$, the desired beam parameters set by the operator $\boldsymbol{b}'$, the beam entering the EA I and the magnet misalignments M. Using the function F, the current beam parameters on the diagnostic screen $\boldsymbol{b}_t$ can be computed from the previously named state components. Ideally, the state $\boldsymbol{s}_t$ would be the same as the observation $\boldsymbol{o}_t$. As is the case with many real-world systems, however, the state of the EA is only partially observable. Both the incoming beam I and the magnet misalignments M are neither measurable nor constant. Instead, only the current actuator values $\boldsymbol{x}_t$, the current beam parameters on the screen $\boldsymbol{b}_t$ and the desired beam parameters $\boldsymbol{b}'$ can be observed. We therefore derive a partial observation $\boldsymbol{o}_t = \left(\boldsymbol{x}_t, \boldsymbol{b}', \boldsymbol{b}_t\right)$.

The most important component of any RL problem definition is the **reward**. The choice of reward is crucial for an RL algorithm's ability to learn. We choose to define the reward function R based on the objective function O. This is convenient as it allows staying close to the original optimisation problem definition. We define the reward as the difference – intuitively the improvement – of the objective function O from time step t to time step $t+1$. This definition is intuitive and does ensure the RL agent learns to minimise the objective. Learning, however, is slow and it is possible for agents to traverse potentially dangerous states of high objective in between two states of low objective during the optimisation. Both these issues are caused by the fact that agents can recover most of the reward lost when traversing states of high objective value in between two states of approximately equal value. Rather than defining constraints to solve this issue, which has to be redone for each problem and may be difficult, the reward is slightly redefined such that the agent is discouraged from worsening the objective at any time unless the expected cumulative reward following such an action is very large. Whenever the objective difference is negative, i.e. the new actuators $\boldsymbol{x}_{t+1}$ result in a higher objective O than the previous actuators x_t, the reward is multiplied by two, making reward recovery much more difficult. We have found that this improves the training speed. The final reward function is defined in terms of an objective function $O(\boldsymbol{x})$ as

$$R(\boldsymbol{s}_t, \boldsymbol{a}_t) = \begin{cases} \hat{R}(\boldsymbol{s}_t, \boldsymbol{a}_t) & \text{if } \hat{R}(\boldsymbol{s}_t, \boldsymbol{a}_t) > 0 \\ 2 \cdot \hat{R}(\boldsymbol{s}_t, \boldsymbol{a}_t) & \text{otherwise}. \end{cases}$$

with $\hat{R}(\boldsymbol{s}_t, \boldsymbol{a}_t) = O(\boldsymbol{x}_t) - O(\boldsymbol{x}_{t+1})$. While the objective function is often already defined for optimisation problems, we found that it is ill-advised to use objectives well suited to classical optimisation as a basis for a reward function. This is because typically used objective functions such as mean squared error (MSE) result in vanishing rewards near the optimum. Over the course of a single optimisation, the reward can change multiple orders of magnitude, especially with common objective functions like MSE. As a result, the reward signal in later stages of optimisations is barely strong enough to facilitate learning in an RL context. This issue is illustrated in Figure 3. Resulting trained agents tend to optimise toward the optimum but stop prematurely. The problem can be addressed by taking the logarithm of the objective function previously chosen for classical optimisation. In the case of the EA optimisation task, we start with a weighted sum over the absolute differences between

[1] Magnet hysteresis occurs when the ferromagnetic core of an electromagnet retains some magnetisation when the current in its coils is removed, reduced or reversed.



(a) MSE (b) Logarithmic

Figure 3. 1-dimensional example of a typically used MSE objective compared to the logarithm of an MAE objective.The MSE differential objective shrinks as the input variable converges toward the optimum, i.e. the reward vanishes. This effect is not present with the logarithmic objective.

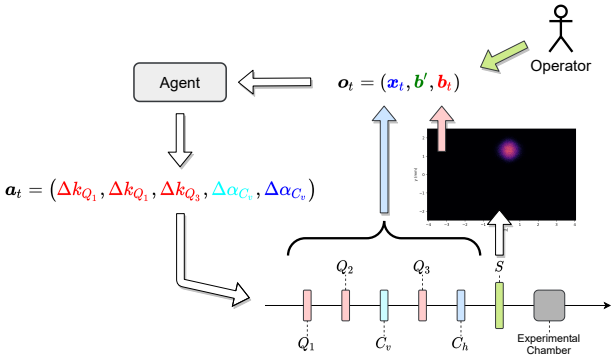


Figure 4. Flowchart of the RL loop as used on the beam positioning and focusing task on the EA. The agent receives as observation from the environment the current actuator values (blue), the beam parameters and intensity (red) as well as the desired beam parameters set by the operator (green). Based on the observation, the agent outputs an action of the change to the settings of the quadrupoles (red), vertical corrector (cyan) and horizontal corrector (blue).

observed and desired transverse beam parameters on diagnostic screen S. Taking the logarithm of the objective results in

$$O(\boldsymbol{x}_t) = \ln \sum_{p \in \boldsymbol{b}_t, p' \in \boldsymbol{b}'} w_p \left| p - p' \right|$$

where w_p is the weight for beam parameter p and a tunable hyperparameter during training of an agent.

The **environment** contains the problem setup and dynamics and is defined in terms of the function $F_{I,M}$ as

$$\mathscr{E}_{\boldsymbol{s}_t}(\boldsymbol{a}_t) = \boldsymbol{o}_{t+1}, r_{t+1} = \left(\boldsymbol{x}_{t+1}, \boldsymbol{b}', F_{I,M}(\boldsymbol{x}_{t+1})\right), R(\boldsymbol{s}_t, \boldsymbol{a}_t).$$

An overview of the RL loop specific to the task in the EA is shown in Figure 4.

4.3. Training

Training in the real world is difficult and dangerous for a variety of reasons. Firstly, training using the real machine can be expensive as a result of limited availability. Beam time on particle accelerators, for example, is a sought-after resource due to how few of these machines exist. This issue is further amplified by the number of samples required for training an RL agent and the time it takes to acquire these samples. The agents trained in this contribution took 6 million samples to train with each sample taking upwards of 7 seconds to run on the real facility. The result is infeasible training times of at least a year of continuous beam time on the real accelerator. Research accelerators are also usually one of a kind, prohibiting the advantages of policy gradient RL algorithms like Proximal Policy Optimisation (PPO) (Schulman et al., 2017) that make use of parallelisation. While the optimisation task on the EA is a particularly strong example for these constraints, they are nonetheless commonplace in the real world. Secondly, it is inherent to RL training that the agent explores unexplored and sometimes extreme regions of the optimisation space. Doing this on a real machine can be quite dangerous, resulting in machine protection systems stopping training in the best case and serious damage in the worst.

To address the issues of availability, speed and safety, we choose to train in a simulation of the accelerator and only deploy the fully trained agent to the real world. We have developed our training setup to facilitate this by splitting the logic of the problem environment in two parts: frontend and backend. The frontend provides the general logic of the problem and handles interaction with the agent. The backend is interchangeable and handles the machine dynamics. This way, it is possible to implement a backend that interfaces with the real machine as well as another that interfaces with a simulation of the machine. By moving the latter functionality to a backend, it is made invisible to the trained agents, which can seamlessly be transferred from simulation to the real machine.

Ideally, the simulation would perfectly represent the real world. Unfortunately, some aspects of the real accelerator, such as the beam I entering the EA or the construction errors M are not known. The incoming beam might even change during operation. For agents to transfer well from training in simulation to running in the real world, it is therefore important to train agents to generalise over these sources of uncertainty. To achieve this, we consider I and M uniformly distributed random variables over reasonable ranges derived from domain knowledge during training in simulation. In each training episode we sample both random variables such that the agents learn to generalise over them. In general, no perfect model exists for any real-world application. It is therefore important to include such errors in the simula-

Table 1. Hyperparameters used to train the RL agents.

Parameter	Value
Action Noise Scale	0.1
Replay Buffer Length $\|R\|$	600 000
Discount Factor γ	0.55
Learning Rate (Actor & Critic)	0.001
Warmup Steps	2000
ANN Architecture	[64, 32]
Total Number of Timesteps	6 000 000
Episode Timestep Limit	50
Beam Parameter Weights	(1, 1, 2, 2)

tion during training as much as is possible. Randomising such potential errors from the real-world during training in simulation also positively impacts the robustness of the final trained agents. As the particular optimisation problem considered in this contribution is about achieving a specific goal that can dynamically be chosen by a human after the agents were trained, the desired beam parameters were also randomly sampled from uniform distributions over ranges derived from the operation of the accelerator. This, too, can be used for other problems with user-defined goals.

In order to accelerate the convergence of the RL training and the optimisation performed by the trained agents, it is important to include domain knowledge where possible. From accelerator theory, it is known that a good beam focus will follow a focus-defocus-focus (FDF) pattern over the quadrupole magnets. We make use of this knowledge in training and subsequent optimisations by initialising the actuators as $\boldsymbol{x}_0 = (10, -10, 10, 0, 0)$ at the start of each episode.

5. Evaluation

The presented implementation has been evaluated on both simulation and machine against baselines, other algorithms as well as human operators. The evaluation was performed on a fixed set of 300 problem instances that each define a sample of the desired transverse beam parameters $\boldsymbol{b}'$, a sample of the incoming beam I, a sample of system errors in the form of transverse misalignments of the quadrupole magnets M and the diagnostic screen S as well as a sample of the initial settings for all five actuators $\boldsymbol{x}_0$. The performance of all solutions is compared in terms of two metrics: MAE and steps. The step metric is defined as the number of function evaluations (set magnets and measure the beam resulting on the diagnostic screen) until the MAE converges, i.e. the change in MAE remains less than the measurement accuracy possible on the real accelerator. The step metric allows for consistency between simulation and real accelerator, where the wall time durations of function evaluations may be very different. The MAE metric is defined as the difference between observed and desired beam parameters

$$\frac{1}{4} \sum_{p \in (\mu_x, \sigma_x, \mu_y, \sigma_y)} |p' - p_t| \tag{1}$$

at the time step t when the beam optimisation is converged.

Three instances of the final RL agent were trained. All RL agents were trained using a benchmarked implementation of the Twin Delayed DDPG (TD3) algorithm (Lillicrap et al., 2015) from the Stable-Baselines3 Python package (Raffin et al., 2019). Each agent was trained for 6 million steps using a custom high-speed particle accelerator simulation (Stein et al., 2022). Training each agent on a cluster node with an Intel Xeon Gold 5115 CPU, two NVIDIA V100 GPUs and 376 GB of RAM took around 14 hours. For the training, RL environments were implemented as *OpenAI Gym* (Brockman et al., 2016) environments around the particle accelerator simulation code backend. Hyperparameters for the TD3 algorithm, the ANNs as well as the environment were selected via a hyperparameter exploration in simulation using *Weights & Biases* (Biewald, 2020) for experiment tracking. The best observed hyperparameters where used to train the models for the final benchmarks. The final hyperparameters are listed in Table 1.

The trained agents were evaluated both on simulation, in order to establish a performance baseline and compare it to alternative algorithms, as well as on the real machine, in order to verify that the results from simulation transfer to the real world and compare the RL agents' performance to that of experienced human operators. An unweighted MAE is used as the evaluation score to compare methods and provide a result that is nicely interpretable as the deviation of the beam parameters that one can expect of the final achieved solution. Table 2 shows the results of the evaluation.

For the comparison in simulation, simple strategies as well as classical optimisation algorithms were evaluated:

- **Do Nothing:** Leave the initial actuator settings as they are. Note that this means that no step is required.
- **Zero:** Turn off all magnets, i.e. setting the actuators vector to $\boldsymbol{x}_n = (0, 0, 0, 0, 0)$, effectively transforming the EA into a drift section. This strategy requires one step.
- **FDF:** Set the quadrupoles to an FDF pattern and the dipoles to 0, resulting in $\boldsymbol{x}_n = \boldsymbol{x}_0$. This strategy requires one step.
- **Random:** A random search of 100 samples over actuator vectors $\boldsymbol{x}_t$. Finally, the actuator vector with the best MAE objective is recovered.
- **Powell:** Run optimisation using Powell's optimisation algorithm (Powell, 1994).

Table 2. Results of the evaluation. We report median MAEs over all problem instances. For the evaluation on the real machine only 200 problem instance were considered. Convergence is defined as the number of steps after which the absolute change in MAE remains smaller than the measurement accuracy on the real accelerator.

Algorithm	**MAE** Median (mm)	**Convergence** Median (Steps)
Do Nothing	1.122	0
Zero	0.588	1
FDF	0.699	1
Random	0.267	101
Powell	0.259	119
COBYLA	0.105	34
Nelder-Mead	0.007	112
Bayesian	0.081	101
Ours	0.008	7
Ours (Machine)	0.036	12

- **COBYLA:** Run optimisation using the COBYLA optimisation algorithm (Powell, 1994). The parameter ρ_0 is set to 1×10^{-3}.
- **Bayesian:** Run optimisation using the Bayesian Optimisation algorithm (Močkus, 1982). The logarithm of the MAE is used as the objective function. Each optimisation is run for 100 steps.
- **Nelder-Mead:** Run optimisation using the Nelder-Mead Simplex optimisation algorithm (Nelder & Mead, 1965). The initial simplex is chosen to span ± 5 for the quadrupoles and $\pm 5 \times 10^{-3}$ for dipoles around $\boldsymbol{x}_0$.

Proven implementations of the above optimisation algorithms from the *scipy* (Virtanen et al., 2020) and *scikit-optimize* (Head et al., 2021) packages were used for this evaluation. Unless stated otherwise, all optimisations use MSE as the objective function and are initialised at $\boldsymbol{x}_0 = (10, -10, 10, 0, 0)$.

Our solution outperforms baselines by up to two orders of magnitude in achieved MAE. Furthermore, the RL agents achieve better MAEs than almost all evaluated classical optimisation algorithms, most of which have previously been demonstrated to perform well on particle accelerator optimisation. While allowed 30 steps to attain the reported MAEs, RL agents manage to converge on an optimal solution in just 7 steps on average. This is significantly less than the steps required by classical optimisation algorithms. In particular, the number of steps to convergence by the RL agents are much less than those taken by the Nelder-Mead Simplex algorithm – the only algorithm to achieve MAEs competitive with our RL solution. A comparison of the convergence of our solution and Nelder-Mead optimisation on the real accelerator is shown in Figure 5. In contrast to other optimisers, the RL agents take a reliably straight path toward their final solution. This reduces the chance of entering potentially dangerous states during the optimisation.

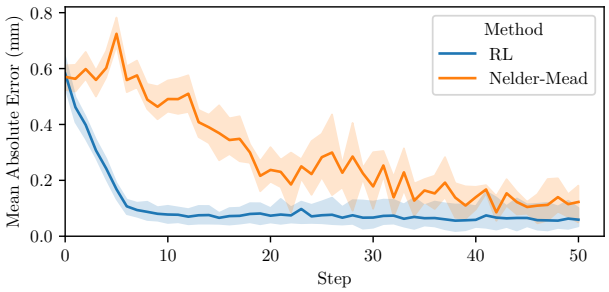
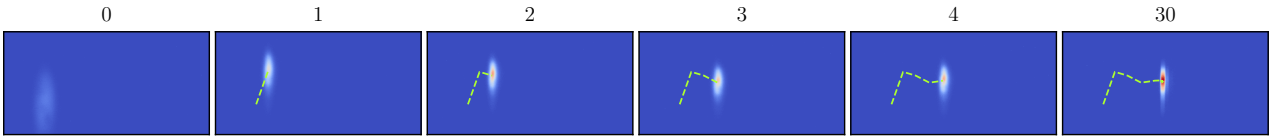


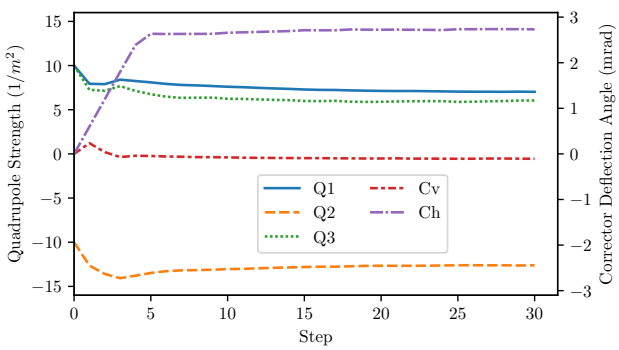
Figure 5. Mean MAE and 95 % confidence interval over 50 steps of three optimisations of three different problem instances by an RL agent and Nelder-Mead optimiser running on the real machine.

Thanks to the training setup described in Section 4.3, trained agents can easily run on the real accelerator by exchanging the backend of the environment. In order to compare the results on the real accelerator with those from simulation, the agents were evaluated over the same desired beam and initial magnet settings as in the evaluation on simulation. Because it is not possible to set the misalignments of the magnets or directly define the beam entering the EA, misalignments and incoming beams were not varied for the experiments on the real accelerator as they were in simulation. Furthermore, it was not possible to evaluate all 300 problems with all three agents due to limited availability of beam time. The evaluation on the real accelerator was therefore limited to 200 problem instances for each evaluated agent. As shown in Table 2, the agents' performance on the real accelerator degrades from a median MAE of 0.008 mm in simulation to 0.036 mm in the real world. This result is still very good and primarily the result of limited measurement accuracy. The length of the edges of the area covered by one camera pixel on the diagnostic screen is 0.013 mm. This pixel size marks the limit of what can be measured in the real world assuming there is no noise. The MAE achieved on the machine is also better than all other algorithms except for Nelder-Mead performed in simulation. It can therefore be concluded that our method transfers well to the real world. Data from an example run on the real accelerator is shown in Figure 6.

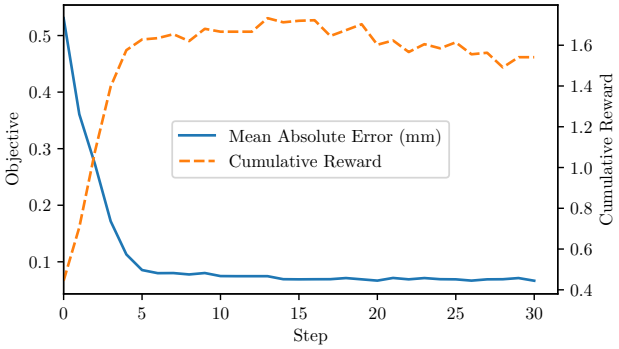
In order to test the effects of machine drifts on agent performance, three problem instances were evaluated on three different occasions over the course of a few weeks. We observed a mean MAE of 0.047 mm and a standard deviation of 0.014 mm. The observed variance is close to the available measurement accuracy. It can therefore be concluded that our RL agents are robust with regards to machine drifts.



(a) Cropped diagnostic screen image at different steps with high beam intensity in red, low intensity in blue and medium intensity in white.



(b) Actuator values over the optimisation run.



(c) Objective and cumulative reward over the optimisation run.

Figure 6. Example optimisation by an RL agent on the real accelerator. The desired beam parameters were set to $\boldsymbol{b}' = (0, 0, 0, 0)$.

Furthermore, the RL agents have been evaluated against two experienced human operators on a problem instance in order to establish the agents' performance in relation to human-level. Note that in this particular comparison, time is measured in wall time instead of steps. This is because human operators do not perform discrete steps in the same way that RL agents have to, as human operators can infer the beam's behaviour before magnets reach requested settings and a clean measurement of the beam is taken. By the end of their optimisation, the human operators achieved a median MAE of 0.039 mm compared to 0.060 mm achieved by the RL agents. The MAEs of both the human operators and RL agents are shown in Figure 7. While the final MAE achieved by the operators is more than the available measurement accuracy better than that achieved by the RL agents, it took the human operators on average 18.0 min to converge on their solution. This is significantly longer than the 1.1 min taken by the RL agents. Note that the key reason for this is fine adjustments toward the end of the operators' optimisations. Over the first 5 min the operators MAE runs only slightly higher than that of the RL agents. While the three agents converged consistently, there was a significant variance in MAE over the first few minutes between the human operators. The observed variance can largely be attributed to different optimisation strategies used by the operators. Unlike the RL agents, which adjust all five actuators at the same time in a direct path toward the optimum, the human operators split the task into multiple sub-tasks. The operators, for example, start by fixing either position or focus of the beam first. We also observed human operators adjusting magnets one after the other. In contrast to the RL agents, operators also limited themselves to only using two of the three quadrupoles in the final solution, presumably to simplify the task. Both final solution patterns are considered physics textbook strategies. Note that on the real accelerator, RL agents spent most of their time waiting for the function evaluation to finish as the magnets in this accelerator section are controlled by relatively slow power supplies. When faster power supplies are available, the RL agents can achieve a much more substantial speed advantage over human operators.

A further experiment was conducted over three problem instances to verify that the performance differences of the Nelder-Mead optimisation algorithm and our RL solution hold on the real accelerator. The resulting MAE curves are shown in Figure 5. After 50 steps, our solution achieved a median MAE of 0.041 mm compared to 0.137 mm achieved by the Nelder-Mead optimiser. Note that as a result of limited beam time, the Nelder-Mead optimisation was given less time on the real machine than it was in simulation, likely explaining the worse result. Nevertheless, the also much faster convergence of our solution is clearly visible in Figure 5. Furthermore, our solution exhibits smoother convergence and smoother changes to the magnet settings than the Nelder-Mead optimisation does. In the real world, smooth actions on the magnets are desired because changing the direction in which magnet settings are changed causes magnet hysteresis. As a result of magnet hysteresis, returning to previous magnet settings after changing them will not result in the same beam on the diagnostic screen and the objective function for the optimisation problem is no longer static. Optimisation algorithms such as Nelder-Mead can struggle with this effect. It was also observed that on the real accelerator, steps taken by Nelder-Mead took an average

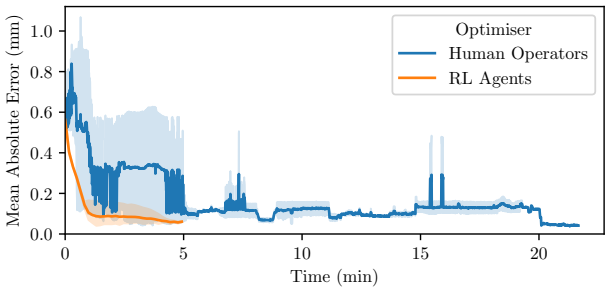


Figure 7. MAE mean and 95 % confidence interval of three RL agents competing against two experienced human operators on the real accelerator in optimising the same problem instance.

of 19.2 s compared to 9.8 s for steps of the RL agent. This observation, too, can be explained by the smooth and direct actions of the RL agents when compared to Nelder-Mead. The RL agents usually move mostly directly to some magnet settings. In doing so, the polarity of the magnet power supplies is usually only changed on the very first step taken and then left unchanged. Nelder-Mead on the other hand changes the polarity of the magnets more often. Doing so in the real world takes a lot longer than changing between settings of the same polarity, resulting in longer average wall time per step.

6. Conclusion

In this contribution, we have successfully trained and deployed an RL agent to a continuous, partially observable, highly non-linear optimisation problem on a real-world machine, without training on the real machine. It has been demonstrated that our RL solution achieves results similar to those achieved by the best alternative optimisation algorithm. Our solution attains these results more than 10 times faster than the only competitive optimisation algorithm. Furthermore, our solution requires neither full-observability of the environment nor machine time for training, significantly simplifying its implementation and reducing the risk of damaging the machine. It has also been shown that despite being trained entirely on a simplified high-speed simulation of the real machine, our solution performs well in the real world thanks to the introduction of domain randomisation over real-world errors during training, achieving results close to the available measurement accuracy as well as those results achieved by experienced human operators. These optimisation results are obtained within the same time or less than it takes experienced human operators to achieve them. The hardware of the specific real-world machine considered in this contribution is relatively slow, meaning that on other problems, an RL optimisation agent is likely to significantly outperform human operators in terms of speed.

Our contribution extends on previous work on RL for the operation of particle accelerators in that the problem is both highly-nonlinear, not fully-observed and yet has a comparatively high-dimensional action space. We also achieve optimisation faster than many previous works.

As is, our solution and similar implementations for other accelerator-related optimisation problems can already be deployed to the control rooms of particle accelerators and decrease the time required for setup and tuning. With less time spent on attaining desired configurations, more time will be available for productive use of these highly advanced but scarce machines. Ultimately, this will improve access to particle accelerators for researchers all over the world and enable high-stakes advances in fields such as physics, material sciences and medicine as well as make the use of accelerators in applications such as cancer treatment and manufacturing more feasible.

In the future, we plan to tune the performance of RL on particle accelerator optimisation problems further as well as extend their application to increasingly complex problems. Not least of all, the goal of future work should be to drive performance beyond the levels of human operators in order to push the limits of operability of particle accelerators, improving sample efficiency in order to reduce the time required for training as well as applying RL to the optimisation of increasingly higher-dimensional problems.

We believe that the experience gained on solving a particle accelerator-related problem using RL as an optimisation agent in this contribution will help drive the application of capable RL solutions to other real-world problems.

Acknowledgements

This work has in part been funded by the IVF project InternLabs-0011 (HIR3X) and the Initiative and Networking Fund by the Helmholtz Association (Autonomous Accelerator, ZT-I-PF-5-6). The authors thank the ARES team Florian Burkart, Willi Kuropka, Hannes Dinter, Frank Mayet, Sonja Jaster-Merz and Thomas Vinatier for their great support during shifts as well as always insightful brainstorms. Furthermore, the authors acknowledge the support from their Helmholtz AI Autonomous Accelerator project partners at the Karlsruhe Institute of Technology (KIT) Andrea Santamaria Garcia, Chenran Xu and Erik Bründermann. In addition, the authors acknowledge support from DESY (Hamburg, Germany), a member of the Helmholtz Association HGF, as well as support through the Maxwell computational resources operated at DESY.

References

Badia, A. P., Piot, B., Kapturowski, S., Sprechmann, P., Vitvitskyi, A., Guo, Z. D., and Blundell, C. Agent57: Outperforming the Atari Human Benchmark. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.

Biewald, L. Experiment tracking with Weights and Biases, 2020. URL https://www.wandb.com/. Software available from wandb.com.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. OpenAI Gym, 2016.

Bruchon, N., Fenu, G., Gaio, G., Lonza, M., O'Shea, F. H., Pellegrino, F. A., and Salvato, E. Basic Reinforcement Learning Techniques to Control the Intensity of a Seeded Free-Electron Laser. *Electronics*, 9, 2020.

Degrave, J., Felici, F., Buchli, J., Neunert, M., Tracey, B., Carpanese, F., Ewalds, T., Hafner, R., Abdolmaleki, A., de Las Casas, D., et al. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602, 2022.

Dulac-Arnold, G., Mankowitz, D., and Hester, T. Challenges of real-world reinforcement learning. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

Duris, J., Kennedy, D., Hanuka, A., Shtalenkova, J., Edelen, A., Baxevanis, P., Egger, A., Cope, T., McIntire, M., Ermon, S., and Ratner, D. Bayesian Optimization of a Free-Electron Laser. *Physical Review Letters*, 124, 2020.

Edelen, A., Neveu, N., Frey, M., Huber, Y., Mayes, C., and Adelmann, A. Machine learning for orders of magnitude speedup in multiobjective optimization of particle accelerator systems. *Physical Review Accelerators and Beams*, 23, 2020.

Eichler, A., Bründermann, E., Burkart, F., Kaiser, J., Kuropka, W., Santamaria Garcia, A., Stein, O., and Xu, C. First steps toward an autonomous accelerator, a common project between DESY and KIT. In *Proceedings of the 12th International Particle Accelerator Conference*, 2021.

Hanuka, A., Duris, J., Shtalenkova, J., Kennedy, D., Edelen, A., Ratner, D., and Huang, X. Online tuning and light source control using a physics-informed Gaussian process. In *Proceedings of the 33rd Conference on Neural Information Processing Systems*, 2019.

Head, T., Kumar, M., Nahrstaedt, H., Louppe, G., and Shcherbatyi, I. scikit-optimize, 2021. URL https://doi.org/10.5281/zenodo.5565057.

Irpan, A. Deep Reinforcement Learning Doesn't Work Yet, 2018. URL https://www.alexirpan.com/2018/02/14/rl-hard.html. Last accessed 26th January 2022.

Ivanov, A. and Agapov, I. Physics-based deep neural networks for beam dynamics in charged particle accelerators. *Physical Review Accelerators and Beams*, 23, 2020.

Kain, V., Hirlander, S., Goddard, B., Velotti, F. M., Della Porta, G. Z., Bruchon, N., and Valentino, G. Sample-efficient reinforcement learning for CERN accelerator control. *Physical Review Accelerators and Beams*, 23, 2020.

Kirschner, J., Mutný, M., Hiller, N., Ischebeck, R., and Krause, A. Adaptive and Safe Bayesian Optimization in High Dimensions via One-Dimensional Subspaces. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. In *4th International Conference on Learning Representations*, 2015.

McIntire, M., Cope, T., Ermon, S., and Ratner, D. Bayesian Optimization of FEL Performance at LCLS. In *Proceedings of the 7th International Particle Accelerator Conference*, 2016.

Močkus, J. The bayesian approach to global optimization. In *System Modeling and Optimization*, volume 38, pp. 473–481, 1982.

Nelder, J. A. and Mead, R. A simplex method for function minimization. *The Computer Journal*, 7, 1965.

OpenAI, Akkaya, I., Andrychowicz, M., Chociej, M., Litwin, M., McGrew, B., Petron, A., Paino, A., Plappert, M., Powell, G., Ribas, R., Schneider, J., Tezak, N., Tworek, J., Welinder, P., Weng, L., Yuan, Q., Zaremba, W., and Zhang, L. Solving Rubik's Cube with a robot hand, 2019.

Pang, X., Thulasidasan, S., and Rybarcyk, L. Autonomous Control of a Particle Accelerator using Deep Reinforcement Learning. In *Workshop on machine learning for engineering modeling, simulation and design at NeurIPS 2020*, 2020.

Panofski, E. et al. Commissioning results and electron beam characterization with the s-band photoinjector at SINBAD-ARES. *Instruments*, 5, 2021.

Powell, M. J. D. A direct search optimization method that models the objective and constraint functions by linear interpolation. In *Advances in optimization and numerical analysis*, pp. 51–67. Springer, 1994.

Raffin, A., Hill, A., Ernestus, M., Gleave, A., Kanervisto, A., and Dormann, N. Stable Baselines3, 2019. URL https://github.com/DLR-RM/stable-baselines3.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal Policy Optimization Algorithms, 2017.

Shalloo, R., Dann, S., Gruse, J.-N., Underwood, C., Antoine, A., Arran, C., Backhouse, M., Baird, C., Balcazar, M., Bourgeois, N., et al. Automation and control of laser wakefield accelerators using Bayesian optimization. *Nature Communications*, 11:1–8, 2020.

Silver, D., Huang, A., Maddison, C., Guez, A., Sifre, L., Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M.and Kavukcuoglu, K., Graepel, T., and Hassabis, D. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529:484–489, 2016.

St. John, J., Herwig, C., Kafkes, D., Mitrevski, J., Pellico, W. A., Perdue, G. N., Quintero-Parra, A., Schupbach, B. A., Seiya, K., Tran, N., Schram, M., Duarte, J. M., Huang, Y., and Keller, R. Real-time artificial intelligence for accelerator control: A study at the Fermilab Booster. *Physical Review Accelerators and Beams*, 24:104601, 2021.

Stein, O., Kaiser, J., and Eichler, A. Accelerating linear beam dynamics simulations for machine learning applications. In *Proceedings of the 13th International Particle Accelerator Conference*, 2022.

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., and Abbeel, P. Domain randomization for transferring deep neural networks from simulation to the real world. *arXiv preprint*, 1703.06907, 2017.

Tomin, S., Geloni, G., Zagorodnov, I., Egger, A., Colocho, W., Valentinov, A., Fomin, Y., Agapov, I., Cope, T., Ratner, D., et al. Progress in automatic software-based optimization of accelerator performance. In *Proceedings of the 7th International Particle Accelerator Conference*, 2016.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J., Polat, İ., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P., and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.