DIPLOMARBEIT

Optimal operation of cryogenic calorimeters through deep reinforcement learning

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Masterstudium Technische Mathematik

eingereicht von

Dipl.-Ing. Felix Wagner, B.Sc.
Matrikelnummer 01426449

ausgeführt am Institut of Analysis and Scientific Computing
der Fakultät für Mathematik und Geoinformation der Technischen Universität Wien
in Zusammenarbeit mit dem Institut für Hochenergiephysik
der Österreichischen Akademie der Wissenschaften

Betreuung
Betreuer: Prof. Dr. Clemens Heitzinger
Mitwirkung: Assistant Prof. Dr. Florian Reindl, Priv.-Doz. Dr. Wolfgang Waltenberger,
Prof. Dr. Jochen Schieck

Wien, 16.05.2023 ____________________ ____________________

(Unterschrift Verfasser) (Unterschrift Betreuer)

Kurzfassung

Kryogene Phononendetektoren mit supraleitenden Thermometern erreichen die höchste Sensitivität für leichte Dunkle-Materie-Kern-Streuung in derzeitigen Direct-Detection Dunkle Materie Experimenten. Bei solchen Sensoren müssen die Temperatur des Thermometers und der Vorspannungsstrom in seinem Ausleseschaltkreis sorgfältig optimiert werden, um optimale Betriebsbedingungen zu erreichen. Diese Aufgabe ist nicht trivial und muss manuell von einem Experten durchgeführt werden. In unserer Arbeit automatisieren wir das Verfahren mit Hilfe von Reinforcement Learning in zwei Situationen. Erstens trainieren wir mit einer Simulation der Reaktion von drei CRESST-Detektoren, die als virtuelle Umgebung verwendet wird. Zweitens trainieren wir live mit denselben Detektoren, die im CRESST-Untergrundlabor betrieben werden. In beiden Fällen gelingt es uns, einen Standarddetektor so schnell zu optimieren, dass unsere Methode in der Praxis eingesetzt werden kann. Zum Schluss erörtern wir vortrainierte Modelle, die die Aufgabe ohne Feinabstimmung auf einzelnen Detektoren erlernen können, wenn ein umfangreicher und vielfältiger Datensatz für das Offline-Training zur Verfügung steht. Unsere Methode kann die Skalierbarkeit von kryogenen Detektorsystemen verbessern.

Optimal operation of cryogenic calorimeters through deep reinforcement learning

F. Wagner[1,2,3,a]

[1] Institut für Hochenergiephysik der Österreichischen Akademie der Wissenschaften, A-1050 Wien, Austria
[2] Institute for Analysis and Scientific Computing, Technische Universität Wien, A-1040 Wien, Austria
[3] Center for Artificial Intelligence and Machine Learning, Technische Universität Wien, A-1040 Wien, Austria

Abstract Cryogenic phonon detectors with superconducting thermometers achieve the strongest sensitivity to light dark matter-nucleus scattering in current direct detection dark matter searches. In such devices, the temperature of the thermometer and the bias current in its readout circuit need careful optimization to achieve optimal operation conditions. This task is not trivial and has to be done manually by an expert. In our work, we automate the procedure with reinforcement learning in two settings. First, we train on a simulation of the response of three CRESST detectors, used as a virtual reinforcement learning environment. Second, we train live on the same detectors operated in the CRESST underground setup. In both settings, we accomplish the optimization of a standard detector sufficiently fast for practical applications. Finally, we discuss pre-trained models that could learn the task without fine-tuning on individual detectors, given a rich and diverse data set for offline training. Our method can improve the scalability of cryogenic detector setups.

Contents

1 Introduction

Several types of particle physics experiments require the sensitive measurement of low-energy particle recoils, e.g. direct detection dark matter (DM) searches and coherent neutrino-nucleus scattering. One successful detector concept is that of cryogenic phonon detectors with transition edge sensors (TES), which were used by the CRESST experiment to reach the currently strongest sensitivity to sub-GeV/c^2 DM-nucleus interactions [1]. These consist of a crystal and an attached TES, acting as a thermometer, where both are cooled to $\approx$ 10 mK. A particle recoil in the crystal induces a distinct phonon population, which thermalizes and produces a temperature increase in the crystal and the TES, leading to a measurable signal for recoils with as low energy as 10 eV [2]. Such good sensitivity requires a careful setup of the detectors, effectively the optimization of the bias current lead through the TES and the base temperature of the crystal and TES, which is governed by the current applied to a heating resistor attached to the crystal.

The optimization of these two values is often done by an expert and by hand. After each change to the control parameters, the system needs to reach its new equilibrium state before the sensitivity can be tested. Therefore the optimization can be a lengthy process, taking up to several hours. The objectives of future physics experiments, e.g. the planned CRESST upgrade [3], require the simultaneous operation of several tens, up to hundreds, detector modules. Automating the optimization is a necessary step to achieve this objective and stay within reasonable bounds of the required manual workload.

The optimal parameter settings vary between similarly manufactured detectors, due to fluctuations in the thermal properties of the TES, crystal, and connections, and the true values are often unknown or have large uncertainties. Calculating the optimal values from theory without previously measuring the detector is, therefore, not reliable. The detector's sensitivity is not uniquely determined by the setting of the control parameters but can also depend on an internal state of the TES, namely the fact whether it was

[a] e-mail: felix.wagner@oeaw.ac.at

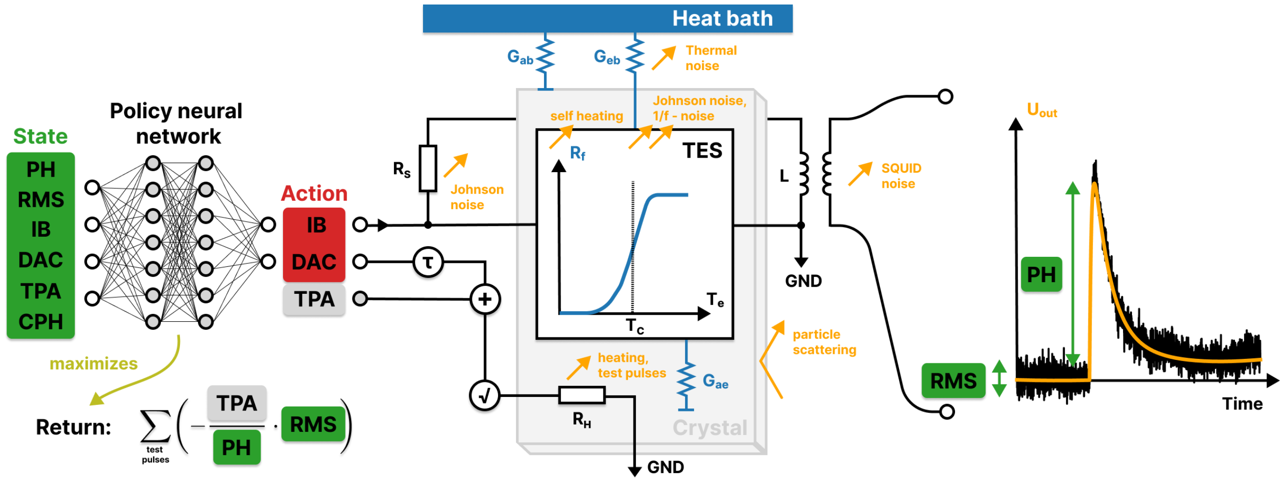

Fig. 1: Schematic drawing of the detector environment. (center) The detector can be described as an electrothermal system, where the readout and heater electronics and the temperatures in the crystal and sensor interact with each other. (right) The recorded observable from particle recoils is a pulse-shaped voltage signal. (left) A neural network is trained with RL to control the measurement settings w.r.t. the signal-to-noise ratio (SNR) of the observed pulses. See text in Sec. 2 and 3 for details.

in a superconducting state before the parameters were set. Therefore simple approaches to optimizing the control parameters, such as a grid search or an educated guess, cannot be optimal in terms of an optimization time to performance trade-off.

In practice, a combination is often used: Choosing a small set of educated guesses for bias currents and recording a one-dimensional sweep of the heating resistor, starting from warm to cold. The operation point (OP), i.e. the final control parameter configuration, is then based on the strength of the detector response. While this approach can work, it has two weaknesses: a) the OP with the strongest detector response is not necessarily the OP that is most sensitive, and b) the sweeps spend an unnecessarily large amount of measurement time in regions of the control parameter space that are unlikely to be optimal, based on the already assembled knowledge from previous observations. While a standard gradient-based or Bayesian optimization approach could improve b) it would also rely on a hard-coded mechanism to approach the desired point in parameter space or risk doing measurements with an unknown internal state of the superconductor.

The previous arguments make it clear that the solution to the optimization problem is not only the set of optimal parameters but also the awareness of an allowed way to approach them, i.e. a sequence. We formulate the problem in the framework of reinforcement learning (RL), a general method to find optimal policies for control problems in discrete time, extensively described e.g. in Ref. [4].

In RL we model the problem as the time-ordered interaction of an agent with an environment. Based on a learned policy, the agent takes actions that depend on its last observation of the environment. The environment returns for each given action a new observation and a reward. The agent's objective is the maximization of returns, which are the sum of rewards over time. The estimated future returns for a given action-observation combination are called values and are learned jointly with the policy. This framework is called a Markov decision process (MDP) if the state satisfies the Markov property.

RL was used to optimize control settings in physics in Refs. [5] and [6] for particle beams, in Ref. [7] for nuclear fusion reactors and in Ref. [8] for superconducting quantum bits.

The state-of-the-art RL algorithms for finding optimal policies in an environment where the actions and observations are continuous values are Actor-Critic (AC) methods, where both policy and value function are approximated with neural networks. Especially the Soft AC algorithm (SAC) performs well in real-world applications, e.g. in robotics [9]. RL methods are typically associated with a low sampling efficiency, i.e. they require many interactions between the agent and environment to discover an optimal policy.

We can mitigate this drawback for our application by using a large experience replay buffer and the excessive training of the agent on previously observed interactions. We provide proof of principle of our method in a virtual environment, modeled after three CRESST detectors operated in run 36 of the experiment and used for a DM search in Ref. [10]. Furthermore, we optimize these three detectors live and directly on the CRESST underground setup by interfacing the experiment control software with our RL agent. Finally, we use the collected replay buffers from the training of our agents in virtual environments to test if a model can be pre-trained to perform the task on new detectors without fine-tuning. This was done similarly in Refs. [11] and [12] for Atari 2600 games and standard control problems, using Transformer models, dubbed Decision Transformers (DTs).

We present the following in this work:

- Our model for the cryogenic detector response and noise contributions is described in Sec. 2.
- We derive a reward function that expresses the goal of a sensitive detector based on observable detector response parameters and explain the approach of optimizing it with SAC agents in Sec. 3.
- We train a SAC agent to optimize a cryogenic detector in our virtual environment in Sec. 4.
- We train and operate on the CRESST setup live in Sec. 5.
- We train a DT model on the collected replay buffers, and test is on unseen, virtual detectors in Sec. 6.

2 Modeling the detector response and noise

The detector's response to particle recoils and other energy depositions depends on the thermal properties of the absorber crystal and TES and the electrical properties of the readout circuit. We can calculate a simplified model of the expected detector response by independently modeling the thermal and electrical circuits involved with ordinary differential equations (ODEs) and solving them jointly as a coupled ODE system. These response calculations were performed analytically for the coupled thermal system of the absorber and TES in Ref. [13] in a small-signal approximation. Analytical calculations for the coupled system of an isolated TES's thermal and electrical response were studied in Ref. [14].

The interacting components are schematically drawn in Fig. 1. The crystal is symbolized by the grey block in the center, the TES by the white rectangle enclosing a sketched graph of the temperature-dependent resistance of the superconducting film $R_f(T)$ that drops sharply around the superconductors transition temperature T_c. The thermal circuit connects the temperature of the heat bath T_b with the temperatures of the crystal (dominated by its phonon temperature) T_a and that of the TES (dominated by its electron temperature) T_e. The heat flow in the system is determined by the thermal connectivity between the components G_{ae}, their individual links to the heat bath G_{eb} and G_{ab}, and the heat capacities of the absorber C_a and TES C_e. The heat capacity of the TES increases below its superconducting transition by a factor of 2.43, see e.g. Ref. [15]. The TES is operated in an electrical parallel connection with a shunt resistor R_s and a readout coil with inductivity L. The circuit is biased with a current I_b from a current source. We neglect the temperature dependency of all properties other than the TES resistance and heat capacity, which provides us with a tractable model for a neighborhood of the critical temperature. The electrical and thermal equations for the state variables are written in Eq. (1). They are coupled through the TES temperature. The system's state variables are the absorber and TES temperatures and the current floating through the TES branch of the readout circuit I_f. They are all time-dependent variables. However, we omit writing their time dependency explicitly for better readability:

$$\begin{aligned}
C_e(T_e)\frac{dT_e}{dt} + (T_e - T_a)\,G_{ea} + (T_e - T_b)\,G_{eb} &= P_e(t),\\
C_a\frac{dT_a}{dt} + (T_a - T_e)\,G_{ea} + (T_a - T_b)\,G_{ab} &= P_a(t),\\
L\frac{dI_f}{dt} + R_s I_b - (R_f(T_e) + R_s)I_f &= 0.
\end{aligned} \tag{1}$$

The system responds to power inputs in the absorber P_a and thermometer P_e, which are introduced by deposited energy ΔE from particle recoils in the crystal. Additionally, a constant current is applied to the heating resistor R_H controlled by the digital-analog converted (*DAC*) value from a digital control system. The *DAC* has a value range between zero and ten and interpolates between no heating and the maximal heating current I_H.

The heater system can also induce heat pulses with a given test pulse amplitude (*TPA*) between zero and ten for testing the detector response. The test pulses decay on an exponential time scale τ_{TP}. The controlling values for the heating current are summed and square rooted, such that they linearly control the power inputs. A particle recoil produces an initial population of athermal phonons from which a share ε thermalizes on a time scale τ_n in the TES and a share $(1-\varepsilon)$ in the absorber, mostly by surface scattering. Assuming an exponential time scale for thermalization is equivalent to assuming a monochromatic athermal phonon population, which is a satisfying approximation for our purposes.

Additional heat input in the system is from the self-heating of the TES, with power $R_f I_f^2$. This contribution is crucial, as it strongly influences the TES temperature and introduces an internal state in the system. The equilibrium state differs depending on whether the TES is in a fully superconducting state. The heat inputs are summarized in Eq. (2) and (3), where we introduced factor δ and δ_H to balance the power inputs from the constant heating and test pulses between the TES and absorber. These factors absorb the spatial dependence of the temperatures, which are non-homogeneous across the system's geometry for the locally induced power from the heating resistor. Furthermore, they absorb the potentially different energy distributions of produced phonons in the constantly applied heating and the fast heater pulses. The factors ε, δ, and δ_H are all values between zero and one. The power input from athermal phonon thermalization is in good approximation uniform across the geometry of the components, as they spread ballistically across the system on a much shorter time scale than they thermalize.

Geometric effects of the temperature distribution in the TES were studied in Ref. [13], where it was shown that such effects could be absorbed in the thermal parameters of the system:

$$P_e(t) = \varepsilon \frac{\Delta E}{\tau_n} \exp\left(\frac{t}{\tau_n}\right) + R_f I_f^2 + \delta \frac{TPA}{10} \exp\left(\frac{t}{\tau_{TP}}\right) + \delta_H \frac{DAC}{10} R_H I_H^2, \quad (2)$$

$$P_a(t) = (1-\varepsilon) \frac{\Delta E}{\tau_n} \exp\left(\frac{t}{\tau_n}\right) + (1-\delta) \frac{TPA}{10} \exp\left(\frac{t}{\tau_{TP}}\right) + (1-\delta_H) \frac{DAC}{10} R_H I_H^2. \quad (3)$$

The detector response is distorted by electrical and thermal noise produced in the system, leading to a finite energy resolution. The contributions to the observed noise power spectrum (NPS) can be modeled as stochastic fluctuations of the right-hand side of Eqs. (1). The major noise contributions that we include in our model are:

- The thermal noise, or phonon noise, ΔI_{th}. This noise arises from natural thermal fluctuations along the thermal coupling of the thermometer and bath, and its contribution is often sub-dominant.
- The Johnson noise of the TES ΔI_{Jf} and the shunt resistor ΔI_{Js}. This noise comes from fluctuations in the movement of the electrons through the resistors that typically dominate the NPS for high frequencies. Excess electrical noise was observed in experiments and described in the literature (e.g. Ref. [14]) and can originate from other electrical components, setup uncertainties, or in the TES. We absorb such excess electrical noise by scaling ΔI_{Js} accordingly.
- The noise introduced by the superconducting quantum interference device (SQUID) amplifier ΔI_{sq}, which is used to measure the magnetic field introduced by L, i.e. the final signal that is digitized and recorded. Its contribution is determined by i_{sq}, a constant value and property of the used SQUID system.
- The 1/f noise $\Delta I_1/f$, also called flicker noise. This noise appears across all TES and other devices, and its origin is not fully clarified. It, therefore, cannot be predicted precisely but depends on an empirical scale factor $\Delta R_{f,flicker}/R_{f0}$. In Ref. [16], it was connected to resistance fluctuations of the TES. It dominates the NPS for low frequencies and is the most harmful noise contribution.
- Several characteristic peaks in the NPS are introduced by the power supply voltage at 50 Hz and its harmonics.

Other known noise contributions exist but were omitted due to their sub-dominance or because they are difficult to model. This includes internal fluctuation noise and any noise sources that would arise in the absorber crystal. Furthermore, burst or telegraph noise was repeatedly observed in CRESST devices and could usually be connected to certain regions within the superconductors transition curve.

To acquire useful descriptions of the detector response and noise, we solve Eqs. (1) on two temperature scales independently: the macroscopic scale of observable, individual energy depositions from heating, particle recoils, or test pulses, and the microscopic scale of thermal and electrical fluctuations. The assumption that these scales do not interact with each other too strongly is, in practice, within reasonable boundaries legitimate.

On a macroscopic scale, Eqs. (1) can be linearized and solved analytically in a small signal approximation, where non-linearities of the TES resistance and the temperature dependency of the TES heat capacity are neglected. However, in our approach, we will treat the detector response model as a simulator and therefore do not require explicit formulas for the response. We, therefore, include all described complexity of the system without additional approximations and solve the ODEs numerically with SciPy's *odeint* method, a wrapper of the FORTRAN LSODA solver that is especially suitable for solving stiff ODE systems [17], to calculate the observed pulse shape. On the microscopic scale, the small signal approximation is very well satisfied, and we use it to derive explicit formulas for the observed NPS in a given OP, described in App. A. We use the method described in Ref. [18] to generate colored noise traces with the calculated NPS. Once pulse shape and noise in a given OP and for defined P_e and P_a are calculated, we superpose them and translate them with the known SQUID settings to a small Voltage trace that would be observed in a real-world setup.

The detector response can additionally depend on the trajectory through which an OP is approached. To include this large-scale time dependency, we start trajectories after a reset of the virtual environment from an edge of the parameter space and solve the ODE system continuously with large mesh grid sizes for intervals without energy depositions, and small ones for intervals where signals are simulated.

For the tests reported in Sec. 4 and 6, we adjust all parameters of our simulation to resemble the detector response and noise of three detectors currently operated in run 36 of the CRESST experiment. Data from these detectors was previously used in a spin-dependent DM search in Ref. [10], and they follow the default CRESST-III design: two of the detectors, called Li1P and Li2P, are optimized to collect athermal phonons produced by nuclear recoils within their absorber crystals made of lithium aluminate. The third detector, Li1L, uses a silicon-on-sapphire (SOS) wafer to collect the scintillation light produced by particle recoils in the scintillating target of Li1P. Li1P and Li1L are operated within a joint housing. However, for this work, we will treat

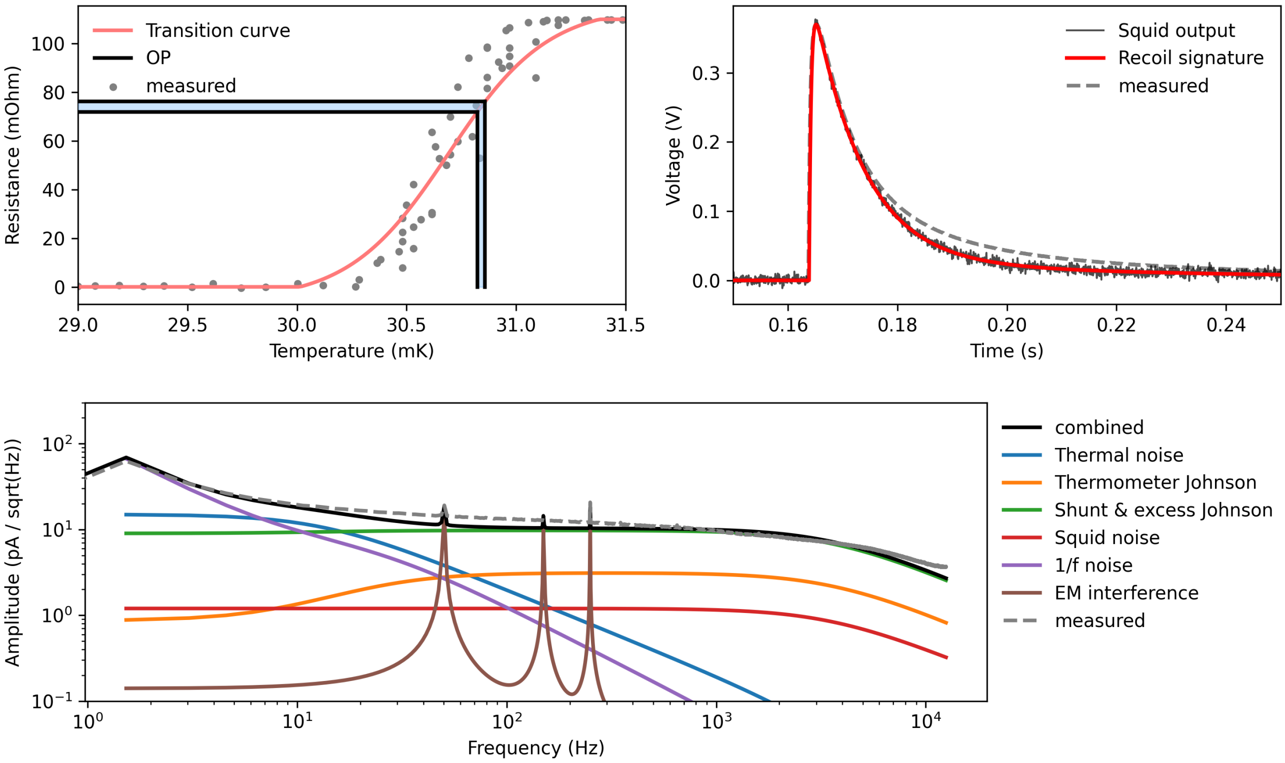

Fig. 2: Simulation and measurement of a 5.95 keV X-ray event induced by a calibration source in the Li1P detector. (upper left) The OP (black/blue lines) within the simulated transition curve of the TES (light red line). A measurement of the transition curve is shown for comparison (grey dots). (upper right) The voltage pulse induced in the simulated SQUID amplifier without noise (red) and overlayed with noise generated from the simulated NPS (black). A measured voltage pulse is shown for comparison (grey dashed). (lower part) The simulated NPS (black) has individual noise contributions (colored). The 1/f, excess Johnson, and EM interference noise components were adjusted to the data. The measured NPS is shown for comparison (grey dashed).

them as independent detectors. We show in Fig. 2 exemplary for Li1P the comparison between measured and simulated transition curve, pulse shape, and NPS, which all agree to a satisfying degree for our purposes. In App. B, the physics parameters extracted from the measurement and used for the simulation are summarized.

2.1 Detector designs with more components

It is straightforward to generalize the ODE system describing a cryogenic detector to designs with more thermal components or TES. Such devices were operated in previous CRESST runs, e.g. composite designs with a separate carrier crystal [19] and mounting structures instrumented with individual TES [1]. Designs with TES separated on a remote wafer were proposed for the operation of delicate target materials [20]. For such non-standard scenarios, the electrothermal system can be jointly written in matrix-vector notation:

$$\begin{aligned}\dot{\underline{T}}(t) = \mathrm{diag}(\underline{C})^{-1} \Big(& \underline{P}\big(t, \underline{T}(t), \underline{I_f}(t)\big) \\ & + \mathrm{diag}\big(\underline{G_b}\big)\big(\underline{T_b} - \underline{T}(t)\big) + \big(\underline{\underline{G}} - \mathrm{diag}(\underline{\underline{G}}\underline{1})\big)\underline{T}(t) \Big), \quad (4)\end{aligned}$$

$$\begin{aligned}\dot{\underline{I_f}}(t) = \mathrm{diag}(\underline{L})^{-1} \Big(& \mathrm{diag}\big(\underline{R_s}\big)\,\underline{I_b} \\ & - \mathrm{diag}\big(\underline{I_f}(t)\big)\big(\underline{R_f}(\underline{T}(t)) + \underline{R_s}\big) \Big), \quad (5)\end{aligned}$$

where underlined (double underlined) quantities are vector (matrix) valued, and $\underline{\underline{G}}$ describes the symmetric matrix of thermal couplings between components. All other quantities are equivalently generalized to vectors from Eq. (1). As we generally neglect noise contributions that do not originate in the TES or the readout circuit directly, our previously introduced description of the detector noise remains unchanged. Designs with multiple TES and heaters are typically harder to optimize as the problem's dimensionality grows with the number of control parameters and observables to optimize. We adjust for our studies in Sec. 4.1 the parameters of Li1P, Li1P, and Li2P such that they resemble a scenario for cryogenic detectors with two TES and correlated heatings.

3 Optimizing the sensitivity

3.1 A metric for sensitivity

The sensitivity of physics searches with cryogenic particle detectors depends mostly on the observable energy bandwidth and the expected signal and observed background rate within this bandwidth. The expected signal and background rates are determined by the chosen target material, mass and shielding, and live time. The energy bandwidth is crucially limited towards its lower end by the energy threshold of the detector, determined by the type of device used and the conditions of its operation. This lower energy threshold E_{th} is for searches conducted with superconducting thermometers, e.g. for light DM, often the crucial value determining the sensitivity. To frame a low energy threshold as an optimization problem, we need to estimate it with easily accessible observables and express its dependency on all controllable quantities, in this case, the DAC and IB values. We derive such a metric in the following, where we use that the sensor noise is a small deflection from an OP and therefore falls in the regime where the height of the observed voltage noise is approximately linear, with proportionality constant γ, to the power of the causing noise source (small signal approximation). Furthermore, the energy threshold of a physics search is usually defined as a multiple of the noise resolution, which can be reasonably well approximated by measuring the observed noise's root-mean-square (RMS) value. Finally, γ is proportional to the inverse ratio of the pulse height PH of a test pulse to its defined input strength TPA.

$$\begin{aligned} &\underset{DAC,\,IB}{\arg\min}(E_{th}) && \Big|\text{small signal approximation} \\ =\, &\underset{DAC,\,IB}{\arg\min}(\gamma U_{th}) && \Big|\text{threshold prop. to resolution} \\ \propto\, &\underset{DAC,\,IB}{\arg\min}(\gamma RMS) && \Big|\text{est. response with test pulses} \\ \propto\, &\underset{DAC,\,IB}{\arg\min}\left(\frac{TPA}{PH}RMS\right) && \end{aligned} \tag{6}$$

We omitted writing the functional dependencies in this derivation for better readability. All appearing quantities depend on DAC and IB, except TPA. The quantity U_{th} is the observed voltage pulse height corresponding to the energy threshold. The assumption of a linear response is in practice not necessarily satisfied, as the transition curve of the superconductor can saturate (flatten) when larger signals are injected. In our objective, this flattening of pulses would be equally penalized as worse noise conditions. By the choice of strength and frequency of injected test pulses we therefore implicitly introduce a trade-off between optimal noise conditions and a linear detector response. For the physics goals of light DM searches we are majorly interested in good performance for small signals, but for practical reasons, e.g. observability of calibration lines, we want to monitor the detector response also for higher energies. Instead of injecting large test pulses less frequently we therefore choose to suppress their relevance for the optimization objective, by introducing a weight factor which we choose as the inverse signal strength $w = 1/TPA$. For usage as a target function in the following sections, we rephrase Eq. (6) to a maximization objective.

$$\begin{aligned} &\underset{DAC,\,IB}{\arg\min}\left(w\frac{TPA}{PH}RMS\right) && \Big|\text{choose } w = \frac{1}{TPA} \\ =\, &\underset{DAC,\,IB}{\arg\min}\left(\frac{1}{\cancel{TPA}}\frac{\cancel{TPA}}{PH}RMS\right) && \Big|\text{phrase as maximization} \\ \equiv\, &\underset{DAC,\,IB}{\arg\max}\left(-\frac{RMS}{PH}\right) && \end{aligned} \tag{7}$$

The adapted target Eq. (7) has several convenient properties:

1. The true strength of the observed pulse is not explicitly contained anymore and the derived function would therefore also work as a target to optimize a detector with triggered pulses from a particle source, where the true energy deposition is not known precisely, instead with test pulses.
2. The function can be evaluated on an event-by-event basis, by measuring the noise RMS in the pre-trigger region of a record window containing a pulse, and by taking the maximum value in the record window as pulse height PH.
3. The target function is always negative and has an upper bound with value zero, which cannot be attained. Furthermore, we can restrict the function to values larger than negative one, as such values can only occur when a record is corrupted by artifacts in the pre-trigger region, e.g. by negative voltage spikes.

Evaluating Eq. (7) for individual events leads to fluctuating values, due to the natural stochasticity of the sensor noise, but it is suitable to be used as a target function in a time-dependent optimization problem.

3.2 Reinforcement learning

RL is a general framework for optimizing time-dependent control problems. The original formulation uses the framework of MDPs, which are defined as a 4-tuple of a state space $\mathscr{S}$, an action space $\mathscr{A}$, a dynamics function

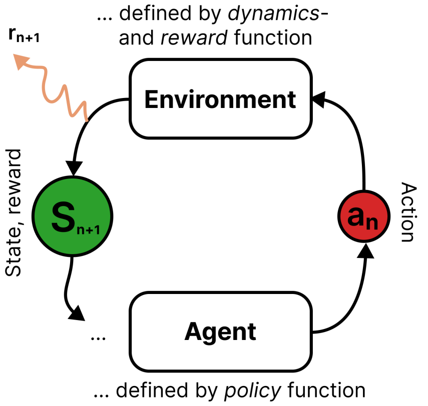
... defined by *dynamics*- and *reward* function
r_{n+1}
Environment
State, reward
S_{n+1}
a_n
Action
...
Agent
... defined by *policy* function

Fig. 3: The mechanics of RL: an agent follows a policy function to interacts with an environment. The environment, defined by its dynamics and reward function, responds to the agent's actions with a reward and observable state. Figure adapted from Ref. [4].

$$p: \quad (A,S) \mapsto \{\text{probabilities for } S'\}, \quad A \in \mathscr{A}, \quad S,S' \in \mathscr{S},$$

which defines state transitions for action-state pairs, and a reward function

$$r: \quad (S,A,S') \mapsto R \in \mathbb{R}$$

that assigns scalars to state transitions. The dynamics and reward function jointly define an environment with observable state, that can be interacted with through actions. The goal of RL is to find a policy function

$$\pi: \quad S \mapsto \pi(A \mid S) = \{\text{probabilities for } A \text{ given state } S\}$$

that maximizes the return R, the sum of collected rewards over time. The policy function is thought of as an agent that interacts with the environment and learns through experience. This framework is schematically summarized in Fig. 3. Dynamics and reward function are a-priori unknown. The definition of an MDP automatically satisfies the Markov property, i.e. p and r only depend on the current state and action and not on prior history. For many practical applications, the state of the environment is not fully observable, and RL literature sometimes accounts for that fact by introducing an observation space separately. We will retain the notation of a state space, but remind us of the fact that the dynamics and reward function of our problem, which we will not further use explicitly, might be more complex than in a simple MDP.

For framing detector operation as a reinforcement learning problem, we define the state and action spaces:

$$\mathscr{S} := \{PH, RMS, IB, DAC, TPA, CPH\},$$
$$\mathscr{A} := \{DAC, IB\},$$

where PH and RMS are the maximum and RMS of the pretrigger region of an injected test pulse, IB and DAC are the set control parameters at the time of recording the pulse and TPA is the strength of the injected pulse. Additionally, we choose to inject after every test pulse with defined TPA a control pulse, a test pulse with maximal signal strength, and include its pulse height CPE in the state. This is done in the CRESST setup per default during detector operation to monitor the position of the OP within the superconducting transition. The values of PH and RMS are directly proportional to the applied bias current. To reduce the complexity of the state space, we divide them by IB and do only allow for positive bias currents. We normalize all action and state values such that they are continuous values within the range $(-1,1)$. We use Eq. (7) as our reward function. The fact that RL maximizes rewards over time implicitly adds the live time of the detector as an additional target, in an equal trade-off with the objectives of a low energy threshold and linear response.

3.3 The Soft Actor-Critic algorithm

We have several requirements for the algorithm we use to train our RL agent. The continuous state and action spaces that cannot be meaningfully discretized require an algorithm that uses function approximators instead of a tabular internal logic. The algorithm should be relatively sample efficient, as the collection of experience in a cryogenic experiment requires $\mathscr{O}$(s) measurement time for each injected test pulse. The initial threshold of finding the superconducting transition in the parameter space requires an algorithm that generally encourages exploration, i.e. the training time spent with the primary objective of collection experience in the environment, in a balance with exploitation, i.e. the time spent with primarily maximizing returns by interpolating the collected experience. The SAC algorithm showed good performance in Ref. [9] in a real-world robotics task with similar requirements, and we therefore chose it for our application.

AC algorithms use – additionally to the policy function π, in this context also called an actor – a value function, or critic

$$q: \quad (S,A) \mapsto Q \in \mathbb{R},$$

which maps action state pairs to estimates of the future return. For the function approximators of both policy and critic we use neural networks π_θ, q_ϕ and train their weights θ, ϕ with gradient descent. For the policy function, we parameterize with the outputs of the neural network a Gaussian function with the dimensionality of the action space, to obtain an explicit conditional probability distribution for actions in a given state. The collected experience is stored in an experience replay buffer from where state transitions (S,A,R,S') are sampled as training data.

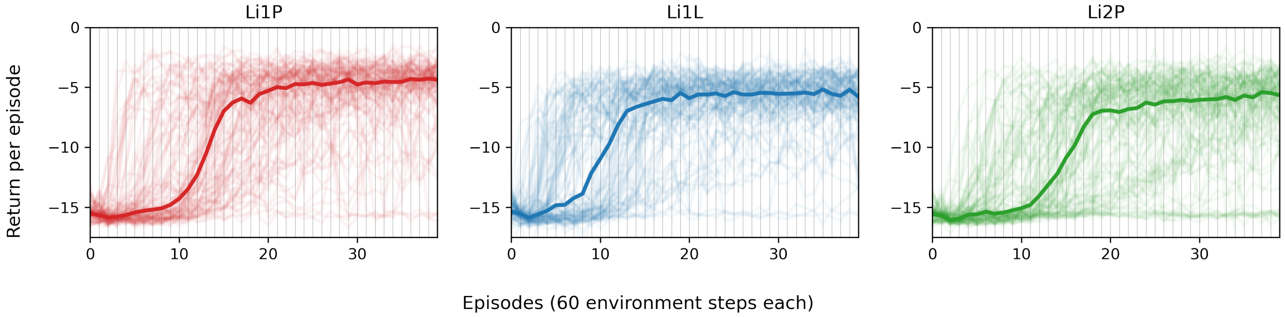

Fig. 4: The returns per episode during training for all 105 versions of the three detectors Li1P (red), Li1L (blue), and Li2P (green). The thick lines are the medians of all curves. The median rises close to the apex of the curves after 15 to 20 episodes. During the first 10 episodes, only little return is collected. The distribution of curves is clearly not normal distributed around the median, which is due to the different phases and hyperparameter settings in the detector versions.

The critic is trained to minimize the soft Bellman residual, with $a' \sim \pi_\phi(\cdot|S')$,

$$J_q(\theta) \propto (q_\theta(S,A) - (R + \gamma(q_{\overline{\theta}}(S',a') - \alpha \ln \pi_\phi(a'|S'))))^2,$$

where the term with the hyperparameter α, called temperature, as coefficient is designed to encourage exploration. The discount factor γ is introduced for numerical stability of long trajectories, and not to be confused with the similarly named factor used for the reward derivation in the previous section. The weights $\overline{\theta}$ of the target critic are discussed later in this section.

The target function minimized by the policy quantifies the Pareto-optimum between exploration and exploitation, with $a \sim \pi_\phi(\cdot|S)$:

$$J_\pi(\phi) \propto \alpha \ln \pi_\phi(a|S) - q_\theta(S,a).$$

There are several technical details of this algorithm that stabilize the training procedure and the exploration-exploitation trade-off:

- Two critics are trained simultaneously, and the minimum of their outputs is used for inference.
- The loss function for the critics makes use of predicted values by their target critics. These are versions of the neural networks with weights $\overline{\theta}$ that are obtained by exponentially smoothing the critic weights θ.
- The value of α is automatically adjusted jointly with the gradient steps done for the neural networks.

SAC is an off-policy algorithm, i.e. the policy function that is learned during training is not necessarily the policy that was used to collect the experience. The fact that data collection and training are two independent processes is useful for practical applications and is exploited in Sec. 5.

4 Operation in a virtual environment

We test the optimization of control parameters with RL in a virtual environment. For this, we wrap the simulation of the three CRESST detectors introduced in Sec. 2 in an OpenAI[1] gym environment and define actions, state, and reward as described in Sec. 3. We send test pulses in random order, with *TPA* values containing all integers from 1 to 10 and the values 0.1 and 0.5. After each test pulse, the agent can adjust the control settings, which jointly represents one environment step. We run episodes of 60 environment steps and reset the detector to a randomly chosen value on the edge of the control parameter space at the start of each episode. One environment step corresponds to the equivalent of 10 seconds of measurement time on the CRESST setup.

We train individual SAC agents for a total of 105 versions of detectors for Li1P, Li1L and Li2P each. All parameters used for the simulation are randomized by 20 % of their original value in each version, where we use the version number as random seed. The training is done for 40 episodes, and we separate the trained versions into three phases of 35 versions each:

1. In the first phase, we do not apply any additional adaptions to the previously explained procedure.
2. In the second phase, we perform a fast sweep of the environment parameters before the training is started and add the collected experience to the replay buffer. This sweep is done by gradually lowering the *DAC* value from its maximal value to zero, while the *IB* value oscillates for each DAC value either from its highest value to zero or vice versa. In total, 120 environment steps are spent in the initial sweep.
3. In the third phase, we study an additional effect that is expected in real-world environments: temperature changes in cryogenics generally take place slowly, as additional components of the structures surrounding the detector might be impacted by the heating on much larger time scales than the observed pulse shapes. The potential impact of this handicap is simulated by delaying the ef-

[1] The main development of this open source project was recently transferred to a fork maintained by the Farama foundation.

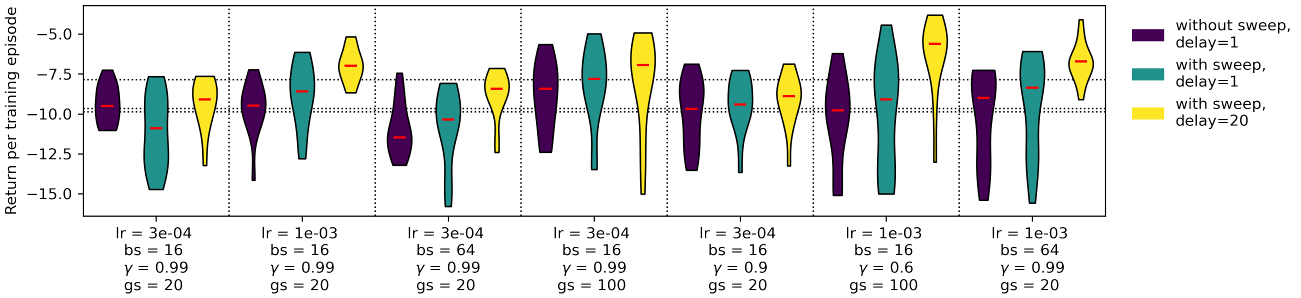

Fig. 5: Returns during training of the different version of the virtual detectors, grouped into violins for each phase (violet, turquoise, yellow) and setting of the hyperparameters learning rate (lr), batch size (bs), γ and gradient steps (gs). Each violin includes five version of Li1P, Li1L and Li2P respectively, sampled and trained with individual random seeds. The red bars indicate the median values of the violins and their thickness the density of the represented return distribution. The dotted horizontal lines indicate the means values of the collected returns of the first (lowest), second (slightly higher) and third (highest) phase of detector versions. The dotted vertical lines separate the violines with different hyperparameter settings. The values for the hyperparameters are written in the ticks on the abscissa.

> fect of the constant heating controlled by the *DAC* value on an exponential time scale of 20 seconds, the equivalent of two environment steps. This delay is also implemented for the other phases, mostly to stabilize the behavior of the numerical ODE solver, but set to a value of 1 second which has no observable impact on the time scale of environment steps.

The resulting return per epoch within the training is shown in Fig. 4 for all versions and detectors. For the large majority of trained detector versions, the return settles on a high value after 15 to 20 episodes, which indicates that an optimal OP is found. This exploitation period is preceded by an exploration period until the agent finds the superconducting transition and a good OP within it.

Within each phase, we iterate through seven different settings for a chosen subset of hyperparameters: the learning rate and batch size for the training of the neural networks, the discount factor γ for the RL training, and the number of gradient steps for which the neural networks are trained on the replay buffer after each environment step. The first setting corresponds to the default setting chosen in Ref. [9] but with a lower batch size and a higher number of gradient steps. Each hyperparameter setting is then repeated for five detector versions and random seeds to reduce the impact of stochastic fluctuations in the training and detector parameters. The results are visualized separately in Fig. 5 for the phases and hyperparameter settings. No significant improvement is visible from the first to the second phase over the duration of the whole training. However, when done only for the first half of the training, the identical plot shows a wider gap between the returns in the first two phases, indicating a small, positive effect of the sweep on the ease of finding the superconducting transition. Optimal OPs are seemingly faster found in the third phase, where the constant heating has a delayed effect, and once found, the collected rewards are also higher. While the second observation is intuitively clear, namely that the randomized exploratory actions of the agent do not lead it as far away from the optimal point as in the other phases, the first observation can be interpreted in different ways. It is e.g. possible that the agent spends more time during the training period, which is dominated by exploration, in certain regions of the control parameter space and can identify them more easily as opportune or unfavorable. More technical details for the implementation and training of the SAC agents are to be found in appendix App. C.1.

In inference trajectories, the agent moves directly to the learned optimal OP. This is in most cases close to the steepest point of the superconductors transition curve, which is for our simplified transition curves at the point of half of its normal conducting resistance. We have observed that the SAC agent moves the chosen control parameters slightly after each pulse. This is likely because multiple OPs in the parameter space are equally optimal, and the agent is not penalized for switching OPs frequently. While this behavior is not generally malicious as long as one of the optimal OPs is fixed for the later period of data taking, it is also not desired.

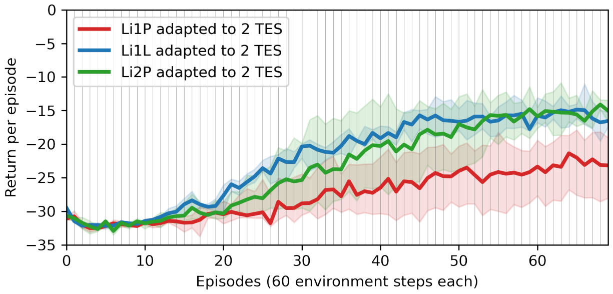

Fig. 6: Return per episode for Li1P (red), Li1L (blue) and Li2P (green) adjusted to two TES. The thick line represents the median of five trained versions of the detectors, sampled with different random seeds. The shaded region shows the upper and lower standard deviations.

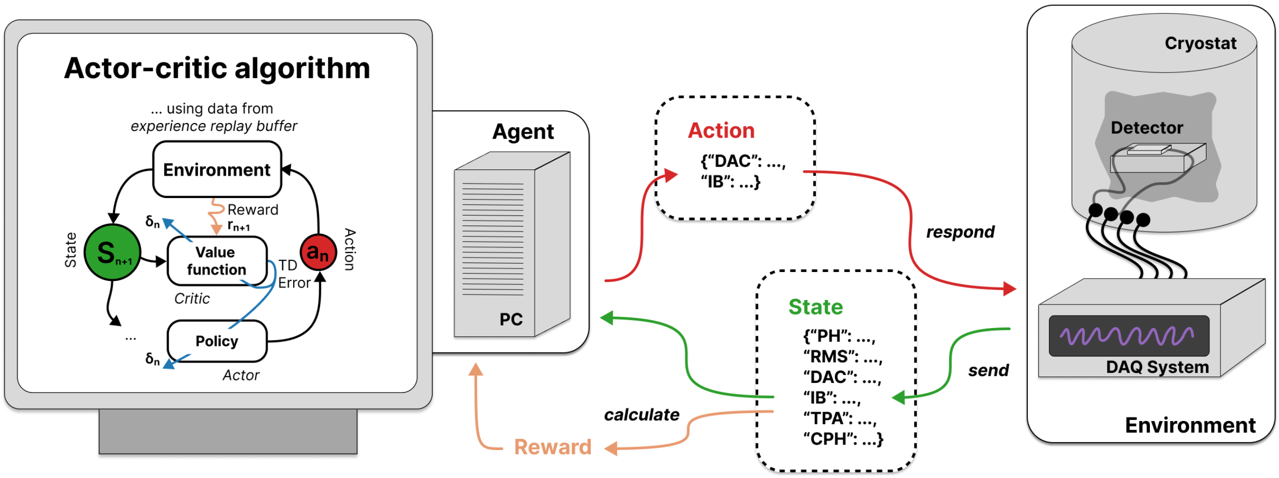

Fig. 7: Schematic visualization of the implemented setup to optimize CRESST detector control. (right side) The detectors are operated in a cryostat and read out by a DAQ system. The parameters of recorded test pulses are sent via an MQTT broker to a client as state. (left side) The client calculates the reward from the state, stores the data in an experience replay buffer, and responds to the DAQ system with new control parameters. An independent process trains the AC agent on the buffer.

We experimented with adding an additional regularisation term to the reward function, which penalizes large steps in the parameter space, by subtracting the Euclidean distance between current and new OP, multiplied by a hyperparameter ω. This regularization did not change the behavior visibly for small values of ω, and had a negative impact on the exploratory behavior of the agent for large values of ω. The regularization was therefore not used in the reported experiments.

4.1 Optimization of multi-component detectors

We investigated how the measurement time required for training scales with the dimensionality of the control parameter space, and the complexity of the detector design. We apply the equivalent procedure described throughout this section to our adaptions of Li1P, Li1L, and Li2P with two TES, described in Sec. 2.1. We use no adaptions to the default training procedure (first phase) and the default (first) set of hyperparameters. The training is repeated for five versions and random seeds and the resulting returns during training are visualized in Fig. 4 for the detectors separately. The training is successful for all versions of Li1L and Li2P, and finding optimal OPs takes roughly twice as long as for the version with a single TES. For the majority of the Li1P versions, only one TES transition is found, and the agent sticks to this local return maximum instead of exploring the environment further. The task is likely harder for the Li1P versions because the superconducting transition curves are closer to the edges of the parameter space, which are less likely to be explored by the Gaussian distributed exploration actions of the agent. In a practical application, this trapping in local optima can be prevented by enforcing more exploration and accepting longer training times.

Overall, it was shown in this section that SAC agents can be trained to find optimal OPs for TES-based cryogenic calorimeters within our virtual environment, for both standard detector designs and such with more observables and control parameters. The required equivalent measurement time varies with the chosen hyperparameters and detector but can be estimated to be several hours. While an expert can likely do this task faster per hand, it is to note that this procedure can be parallelized for all operated detectors, and executed during time periods when manual interactions are cumbersome, e.g. at night. In the following section, we validate our method by operating on the real-world environment of the CRESST experimental setup.

5 Live operation on the CRESST setup

A measurement interval of 12 days in February 2023 was dedicated to testing the method of optimizing detector operation live on the CRESST underground setup in the Laboratori Nationali del Gran Sasso (LNGS). Experiments were performed with the real-world versions of Li1P, Li1L, and Li2P, the three detectors of which virtual twins were described in Sec. 2 and used for RL experiments in virtual environments in Sec. 4. For the first week of the measurement interval, data taking was stopped, and the technical setup was implemented. Normal operation and data taking were continued after this initial week with the other detec-

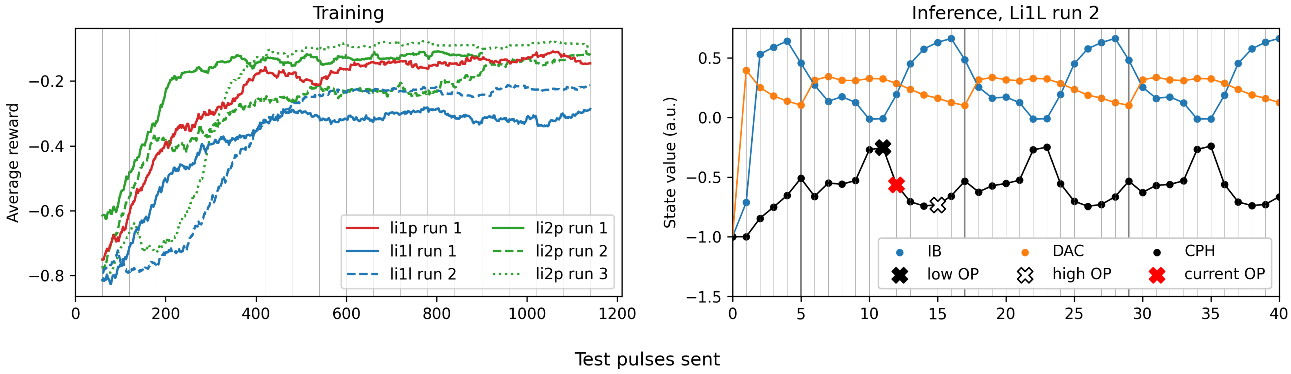

Fig. 8: (left) Average rewards per test pulse sent during the live training on the CRESST setup, smoothed with a moving average of 60 test pulses. Results from six runs with different training settings are shown for Li1P (red), Li1L (blue, blue dashed), and Li2P (green, green dashed, green dotted). (right) Exemplary trajectories of *IB* (blue), *DAC* (orange), and *CPH* (black) were obtained by a SAC agent in inference on Li1L after training run 2. The lines connecting environment steps are a guide for the eye. The thick, vertical grey lines mark the beginning of the periodically injected test pulse strengths, starting from the smallest strength. The agent prefers different OPs for different injection strengths, the lowest (highest) CPH corresponding to the OP chosen highest (lowest) in the transition is marked with a white (black) cross, and an intermediate OP is marked with a red cross.

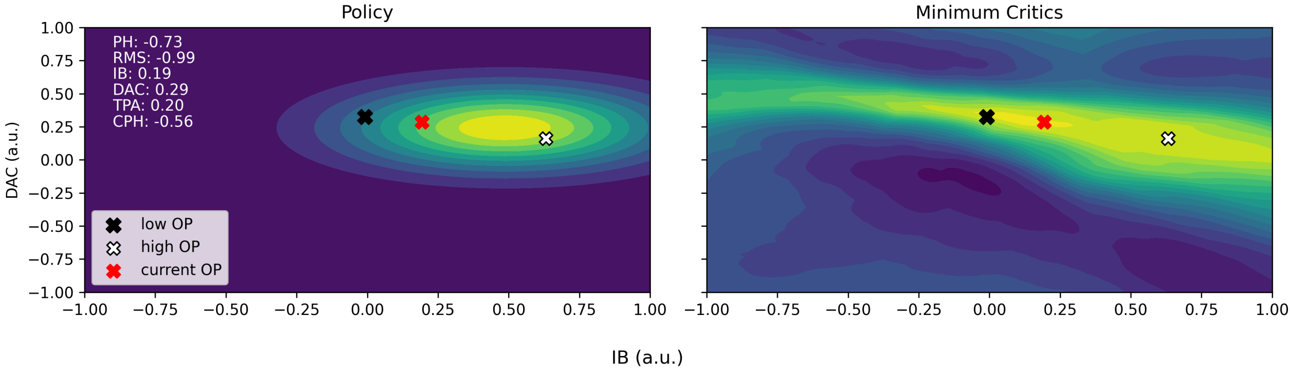

Fig. 9: Contour plot (yellow – high values, violet – low values) of the Gaussian policy (left) and the minimum of the critic functions (right) of a SAC agent trained on Li1L (run2), in a certain state (red cross, white text left) during inference. The same inference trajectories are visualized in Fig. 8, and the black, white, and red crosses correspond to each other.

tor modules in the setup, on which no RL experiments were performed.

On the CRESST setup test pulses are induced in cyclic order, with the same TPA values that we used in the virtual environment. The communication between our Python-based RL infrastructure and the control and data acquisition (DAQ) software of the CRESST setup was realized via messages sent through an MQTT broker. The DAQ system, acting as the RL environment, induces test and control pulses through the heater electronics and records the detector response. Pulse shape parameters are calculated, and an MQTT message is broadcast via the broker and received by the machine on which we run the RL infrastructure. On this machine, we run two parallel processes:

1. The first process receives messages, calculates rewards, and writes data to the replay buffer. A policy model is queried with the state of each received message. The outputs are compiled into a reply containing new control settings.
2. The second process continuously trains the agent with a SAC algorithm on the replay buffer. If the desired number of gradient steps is reached before new data is added to the replay buffer, the process is paused. The accessibility of the replay buffer from both processes is realized through memory-mapped arrays.

This setup is schematically visualized in Fig. 7. We run experiments consecutively as our current implementation of device communication does not support work on multiple channels in parallel. A total of 48 experiments were run with measurement times between one and three hours, where the majority was used for implementation and debugging of the setup, and the final 6 runs were used as performance benchmarks of the method. We made individual adaptions to the hyperparameters, the configuration of the state space, and the number and length of training episodes in all runs. These and further details of the training process are summarized in App. C.2.

One performance run was performed with Li1P, two with Li1L, and three with Li2P. Each run was started with a rough sweep of the action space, as done in phases 2 and 3 of the detector versions in Sec. 4. We have observed delays until strong changes in the heating take place, as modeled in phase 3 of virtual versions. However, the obtained returns in the live setup are not directly comparable to those obtained in the virtual setup due to technical differences in calculating pulse shape parameters. The average rewards obtained during training, depending on the number of test pulses sent since the start, is shown in Fig. 8 (left). In all runs, a high plateau of rewards is reached before 600 test pulses were sent, corresponding to roughly 1.5 hours of measurement time. In comparing the runs, we have qualitatively observed that a richer state space leads to a longer required time until an optimal OP is found, but also to generally better responsitivity to the environment. After training is completed, we run inference trajectories with all trained agents. They all find suitable OPs and feature a similar strategy that is exemplary visualized for Li1L run2 in Fig. 8 (right). The agent adjusts the OP to the TPA value of the injected test pulse expected next in the cyclic test pulse queue. The height of the OP in the superconducting transition curve can be inferred from the height of control pulses. The behavior of cyclically choosing different OPs is additionally visualized in Fig. 9, where the values of both the Gaussian policy and the minimum of the critic functions are shown as contours in the two-dimensional control parameter space for a fixed state. Given the current state, we can infer the position of the superconducting transition in the control parameter space from the values of the critic functions. The negative tilt of the favored IB and DAC combinations observable in the critic functions can be interpreted as the effect of the self-heating that is proportional to IB^2. The policy has learned to move between OPs, visible by the apex of the Gaussian distribution that indicates the next control parameter setting the agent will choose, depending on the TPA value expected to be injected next in the periodically repeating sequence.

In summary, RL is a practically applicable method for finding optimal control parameters for cryogenic TES-based calorimeters in the real world. Our studies from the virtual environment discussed in Sec. 4 generalized well or were exceeded by the results obtained on the CRESST setup. We generally expected the agents to respond to all information contained in the state, including the sent TPA values. The observed responsiveness in inference can therefore be interpreted as a confirmation that the agents adapt to the detectors they are trained on appropriately. For physics data taking, fixing one OP out of the control parameter region favored by the agent is preferable. The time period needed for training is comparable to the time period that is needed by an expert to do the optimization via manual interventions. It is in this setup necessary to train an individual SAC agent for each detector, as the policy weights encode inextricably both the strategy to approach an OP and the knowledge of the physics of the detector at hand that determines the position of optimal OPs in the control parameter space. Possibilities to disentangle this information and shorten the necessary exploratory phase for new detectors are discussed in Sec. 6.

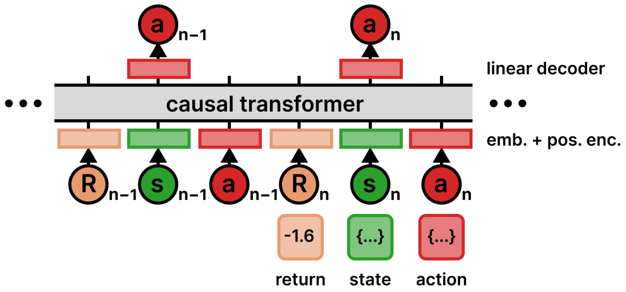

Fig. 10: Schematic visualization of a DT model. Returns (orange), states (green), and actions (red) are individually and linearly embedded into a latent space of fixed shape. Their position in the sequence is jointly encoded. They are then consecutively input to an autoregressive Transformer encoder model that predicts the next item in the sequence, with future items causally masked. Figure adapted from Ref. [11].

6 Towards a universally pre-trained model

The information that has to be learned by the agent by processing the environmental responses can be thought of in three layers:

1. the current state of the detector,
2. the physics parameters of the detector worked at,
3. and a general strategy for optimizing control parameters of cryogenic TES-based detectors.

The state of the superconductor is suitably encoded in the observable state of the RL environment, given the overall knowledge of the expected response from the detectors worked at. An individual state does not contain enough information to learn the physics parameters of a detector, but a suitable sequence of states does. The reward is also required to communicate its decisions' quality to the agent. When we train SAC agents on individual detectors, the physics parameters and general strategy are both encoded in the policy parameters. However, a properly engineered state that accumulates all relevant information from a longer sequence of observations can also contain the information of the physics parameters of the detector, leaving only the general state-depended optimization strategy to be hardwired in the trained policy parameters. This can enable an agent, pre-trained on a diverse set of detectors, that can, during an inference trajectory, combine exploratory actions with exploitation and adapt

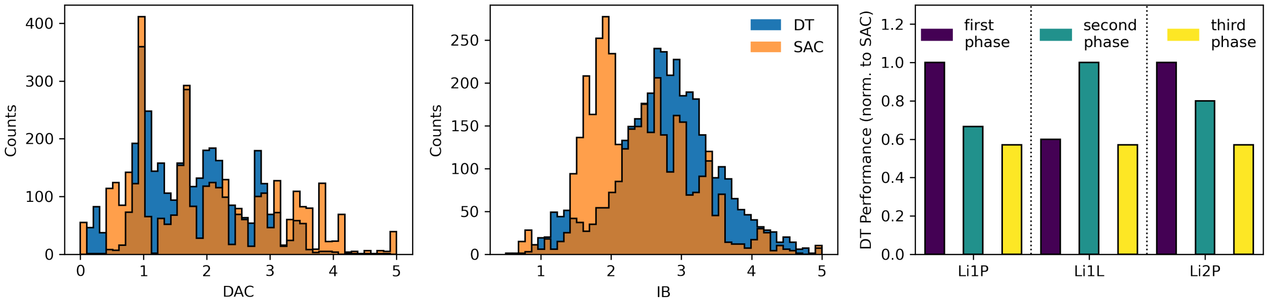

Fig. 11: (left, center) Distribution of the chosen control parameters *DAC* (left) and *IB* (center) by the DT model (blue) and SAC agents (orange) during 21 inference trajectories with Li1P, Li1L, and Li2P detector versions each. (right) A statistic of the performance of the DT model normalized to the SAC agents. The performance is individually stated for the versions of Li1P (left), Li1L (center), and Li2P (right), marked by the ticks on the abscissa and separated by dotted black lines, and the three phases (violet, turquoise, yellow – see also Fig. 5). See the text for a definition of the performance metric.

its strategy by that to new detectors without the need for individual training. Such approaches were studied for other RL environments in Refs. [21–23] by using dedicated network architectures or randomization techniques to control robots that adapt to individual real-world uncertainties in the observed state; and in Ref. [12], where one DT model was trained to reach an expert level on multiple Atari 2600 games and reduce the required time for finetuning to new games. We report in this section on our studies of using a DT model that we train offline on the replay buffers which were collected during the training in virtual environments, reported in Sec. 4.

6.1 Decision Transformers

The Transformer neural network architecture was introduced in Ref. [24]. Since then, it has become the state-of-the-art model for extracting information from and predicting sequences and is widely used in natural language processing. Large pre-trained language models, such as the Generative Pre-trained Transformer (GPT) models [25], have recently started significantly impacting academia, industry, and society.

The Transformer architecture has an encoding and a decoding component. We focus on the encoder, which is relevant to this work. The encoder consists of three major parts, where all functions are trainable neural networks:

- Sequence elements are linearly mapped into a fixed-sized embedding space. Additionally, the position in the sequence is linearly mapped to the same shape and added to the embedded sequence element.
- A self-attention head extracts a composite state containing information from all relevant sequence elements. A self-attention head consists of three linear mappings, constructing key vectors k_i, value vectors v_i, and query vectors q_i from the i'th embedded sequence element. The inner product of the key and query provides a weight factor, quantifying the relevance of the information contained in the keyed sequence element to the querying one. The sum of the weighted values accumulates into a composite state z_i for each sequence element:

  $$z_i := \mathrm{softmax}\left(\frac{q_i K^T}{\sqrt{d_k}}\right) V,$$

  where softmax is the softmax function, K and V are matrices containing the query and keys vectors of all sequence elements as their rows, and d_k, is the number of dimensions of the key and query vectors, providing a practically useful scaling factor.
- This composite state z_i is input to a deep neural network. If multiple self-attention heads are used, their outputs are concatenated and jointly input into the network.

The Transformer encoder can be thought of as a generalization of feature extraction methods, where neighborhood and contextual relations are not hardwired in the network's architecture but learned from the data jointly with the desired logic. Typically larger amounts of data and computing power are needed than for the training of convolutional, recurrent, or graph neural networks.

The RL problem is approached by DT models differently than by AC algorithms, namely by modeling it as a sequence prediction problem. The Transformer is trained offline on trajectories of successive return-to-go, states, and actions. There is no direct interaction with the environment during training, the data has to be collected previously. Return-to-go is the sum of the rewards that will be collected later in the trajectory. The trajectory input to the transformer is causally masked, i.e. future actions, states, and returns are not observable for predicting prior actions. By this, the DT model learns to predict actions that are expected to lead to the given returns in the further trajectory. In inference, the DT trajectory is started with a high but reasonably achievable return for a trajectory in the given environment, and received rewards are subtracted from this initially prompted return while the trajectory evolves through interaction with the environment. In Ref. [11], it was shown that this approach works equally well as AC methods by training on collected data for individual and fixed environments. A generalization to multi-game scenarios, where one model learns to act optimally in multiple environments, was presented in Ref. [12].

6.2 Generalizing to unseen detectors

During the training of virtual detector versions (see Sec. 4), a total replay buffer data set of 756,000 state transitions in the cryogenic detector environment was collected from 315 randomized versions of Li1P, Li1L, and Li2P and distributed to 12,600 training episodes. We split this data set into three parts in the ratio 6/2/2 by accumulating trajectories from 3/1/1 with similar settings but different random seeds sampled and trained detector versions into a training/validation/test set. We trained a Transformer architecture with 40 M parameters that was successfully used in Ref. [12]. We also trained smaller architectures, which reached only worse performance. It is a known phenomenon, which was also studied in Ref. [12], that larger Transformer architectures train faster and scale better with their number of parameters than other neural network architectures. The validation set is used to monitor the training process, and the test set is used to state performance benchmarks. The loss values obtained on all three sets are in a similar order of magnitude, and the validation and test loss are almost identical. The evolution of validation and training loss during the training show similar characteristics, which usually signals that the model does not significantly overfit the data contained in the training set. Details of the models and training are contained in App. C.3.

We run inference trajectories on the detector versions contained in the test set and record the state, action, and reward sequences for the contained 21 versions of Li1P, Li1L, and Li2P each, equally distributed to the three phases of detector versions. In inference, we start the trajectories by providing a return-to-go value to the DT model of -1.59, which is the highest return in the training data set. To compare the performance between the trained SAC models and the DT model, we calculate a performance metric that represents if the model can find OPs in the superconducting transition curve. The performance is quantified by counting the number of detector versions for which an OP in the superconducting transition curve is found, i.e. where at least 10 out of 60 of the chosen actions in the inference trajectory lead to a TES resistance between 10 % and 90 % of its normal conducting value.

Overall, the DT model reaches 75.34 % of the performance of the SAC models. A more refined performance distribution is shown in Fig. 11 (right). A significant effect is not visible between the detector models but between the third phase of training and the first two, where the DT model performs better on the first two. This performance difference is possibly due to the similarity between the first two phases, effectively making generalization easier. Note that SAC agents were trained and tested exclusively on one detector version each, while the DT model had not seen the detector versions in the test set before.

We performed tests to investigate if the Transformer model mostly memorizes information, in this case, if it learns to make good guesses individually for the Li1P, Li1L, and Li2P detector versions, or if it learns the desired logic of decision-making. For this, we plot the distribution of chosen actions, the *DAC*, and *IB* values, of the DT model and the SAC model in Fig. 11 (left, center). The SAC actions are more tightly distributed around certain values, while the DT actions are more spread out. This is expected as the SAC agents simply revert in inference to OPs learned during training, while the DT model has to start inference trajectories generally with exploratory actions to familiarize with the yet unseen detector version. While the distribution of chosen actions, and with that, the distribution of optimal OPs, is not uniformly distributed in the control parameter space, it is relatively widespread and covers almost the full available parameter space. This is an indication that in our data set while being built from only three original detector models, through the randomization procedure, we reach a meaningful diversity in the final data set. In inference trajectories, the DT model appears to use the strategy of initially guessing suitable parameters and then slowly sweeping the heating down until either pulses or a transition in the observed noise is visible. This strategy is also commonly used by human experts but not explicitly contained in the training set, as it requires the simultaneous observation of multiple time steps to work efficiently. We interpret this as an indication that the model does indeed learn an underlying logic to strategically optimize detector control parameters.

However, the current version of the trained model is limited: when an optimal OP is not below the initially guessed *DAC* value, the model usually does not find an OP, and the learned logic is not fully consistent as sometimes, despite the observation of a pulse the model does not steer back to the corresponding region of the parameter space. The model varies the bias current more strongly than the constant heating, which is also done by the SAC agents and can therefore be a learned feature from training data or be because the bias current generally has less impact on the position in the transition curve. Also, the current version of the trained DT model generally obtains lower rewards than the SAC models trained on individual detector versions.

We expect that the performance of the DT model can be significantly improved by a larger data set with more diversity in the contained detector versions, subsequently larger models, and longer training. Furthermore, there are more sophisticated ways initial return values can be prompted, e.g. used in Ref. [12]. However, already the presented version can be useful in practice to find suitable initial values in a search for optimal control parameters on new, unknown detectors.

7 Conclusion

In this work, we presented studies for automating the optimization of control parameters of TES-based cryogenic calorimeters with RL. We simulated the response and noise of three CRESST detector modules operated in run 36 of the experiment and trained SAC agents to find optimal OPs for them. We sampled randomized versions of the detectors and systematically studied hyperparameters of the training and RL setting on a total of 105 different versions of each detector. Furthermore, we test five randomized versions of the three detectors each, adapted to designs with two TES. We tested our method on the real-world versions of the worked-on detectors operated in the underground CRESST setup in 6 representative test runs across the three detectors. In all our experiments, the required equivalent measurement time to complete training was in the order of magnitude of several hours, fast enough for practical usage. The training was successful on all representative runs on the live setup and most runs in the virtual environment. We concluded our studies by training a DT model on a subset of the collected replay buffers dedicated to training to do the task without finetuning on unseen detectors. The model reaches 75.34 % of the individually trained SAC performance on a test set of detector versions, benchmarked by the capability of finding OPs in the superconducting transition curve.

SAC agents and the DT model are two ends of a spectrum of model choices. Intermediate steps would be e.g. AC algorithms with Long Short Term Memory [26] or attention-based policy and value functions. The policy and value networks used for the SAC model have the additional advantage that they can be run on small machines or dedicated hardware for fast inference. The DT model has the advantage that only one model has to be maintained for all detectors, which can be hosted on a central server and queried remotely. Fur future works, the presented method could be combined with more control parameters, e.g. that of an active magnetic field compensation. Furthermore, richer information extracted from the observed ADC signal, e.g. a combination with networks that discriminate pulses from artifacts and pile-up as reported in Ref. [27], could additionally improve the stability and convergence speed of the agents. Stable measurement setups with fast relaxation time can also be used with higher rates of test pulses, reducing the required measurement time.

In summary, the presented method can significantly reduce the required time for the initial control parameter optimization of large multi-detector setups and increase the overall detector live time. Jointly with the previously by the CRESST collaboration published deep learning method for automated data cleaning for cryogenic detectors [27], we can improve the possibilities of scaling up the number of simultaneously operated detectors in future experimental setups.

Acknowledgements This project was a cooperation between the Institute of High Energy Physics of the Austrian Academy of Sciences and the Technical University Vienna. FW was funded by the Austrian research promotion agency (FFG), project ML4CPD. The computational results presented were partially obtained using the Vienna CLIP cluster and partially using the LNGS computing infrastructure. Pulse shape processing was done using the Cait open source package [28]. FW would like to express his deepest gratitude for the support he received during this project and his studies. He extends his thanks to: Franz Pröbst, Vanessa Zema, Florian Reindl, Wolfgang Waltenberger, Jochen Schieck, Clemens Heitzinger, the CRESST collaboration, his study group, friends, family and partner.

App. A: Noise contributions

The here presented derivation of the noise contributions follows Ref. [14], the model of flicker noise Ref. [16]. We consider only the thermal equation of the TES and the electrical equation of its readout circuit, the first and third line in Eq. (1), and ignore fluctuations that might arise from the interaction of absorber and TES. These equations can be linearized in a small signal approximation. Then, equations for the resulting temperature fluctuations in the TES ΔT_e and current fluctuation in the TES branch ΔI_f for small, linear inhomogeneities, i.e. power fluctuations ΔP_e in the TES and voltage fluctuations ΔU in the readout circuit, can be derived. These can be summarized in a transition matrix

$$\begin{pmatrix}\Delta T_e \\ \Delta I_f\end{pmatrix} = \begin{pmatrix} s_{11}(w) & s_{12}(w) \\ s_{21}(w) & s_{22}(w)\end{pmatrix}^{int/ext} \begin{pmatrix}\Delta P_e \\ \Delta U\end{pmatrix},$$

where the matrix elements read differently for noise that has its origin in the TES, dubbed *int*, and in other parts of the readout loop, dubbed *ext*. The relevant matrix elements are

$$
\begin{aligned}
s_{21}^{int}(w) = -\frac{1}{I_{f0}}\Bigg[&\frac{L}{\tau_{\mathrm{el}}\mathscr{L}_{\mathrm{I}}} + \left(R_{f0} - R_s\right) \\
&+ 2\pi \mathrm{i} w \frac{\mathrm{L}\tau}{\mathscr{L}_{\mathrm{I}}}\left(\frac{1}{\tau} + \frac{1}{\tau_{\mathrm{el}}}\right) - \frac{4\pi^2 w^2 \tau \mathrm{L}}{\mathscr{L}_{\mathrm{I}}}\Bigg]^{-1},
\end{aligned}
$$

$$
s_{22}^{int}(w) = -s_{21}^{int}(w) I_{f0} \frac{1}{\mathscr{L}_{\mathrm{I}}}(1 + 2\pi w i \tau),
$$

$$
\begin{aligned}
s_{21}^{ext}(w) = -\frac{1}{I_{f0}}\Bigg[&\frac{L\tau}{\tau_{\mathrm{el}}\tau_I \mathscr{L}_{\mathrm{I}}} + 2R_{f0} \\
&+ 2\pi \mathrm{i} w \frac{\mathrm{L}\tau}{\mathscr{L}_{\mathrm{I}}}\left(\frac{1}{\tau_I} + \frac{1}{\tau_{\mathrm{el}}}\right) - \frac{4\pi^2 w^2 \tau \mathrm{L}}{\mathscr{L}_{\mathrm{I}}}\Bigg]^{-1},
\end{aligned}
$$

$$
s_{22}^{ext}(w) = s_{21}^{ext}(w) I_{f0} \frac{\mathscr{L}_{\mathrm{I}} - 1}{\mathscr{L}_{\mathrm{I}}}(1 + 2\pi w i \tau_I),
$$

where we used the same definitions as in Ref. [14]:

$$
\tau := \frac{C_e}{G_{eb}}, \qquad \tau_{el} := \frac{L}{R_{f0} + R_S},
$$

$$
\mathscr{L}_I := \frac{I_{f0}^2}{G_{eb}} \left.\frac{dR_f}{dT_e}\right|_{T_{e0}}, \qquad \tau_I := \frac{\tau}{(1 - \mathscr{L}_I)}.
$$

The thermal or phonon noise can then be calculated by

$$
\begin{aligned}
|\Delta I_f(w)|_{ph}^2 &= \left|s_{21}^{int}(w)\right|^2 |\Delta P_e|_{ph}^2, \\
|\Delta P_e|_{ph}^2 &= 4k_B T_e^2 G_{eb} \frac{2}{5} \frac{1 - \left(\frac{T_b}{T_{e0}}\right)^5}{1 - \left(\frac{T_b}{T_{e0}}\right)^2},
\end{aligned}
$$

the electrical Johnson noise in the TES by

$$
\begin{aligned}
|\Delta I_f(w)|_{Jf}^2 &= \left|s_{22}^{int}(w)\right|^2 |\Delta U|_{Jf}^2, \\
|\Delta U|_{Jf}^2 &= 4k_B T_e R_{f0},
\end{aligned}
$$

the electrical Johnson noise in the shunt resistor, scaled by a factor E_J to account for excess electrical noise, by

$$
\begin{aligned}
|\Delta I_f(w)|_{Js}^2 &= \left|s_{22}^{ext}(w)\right|^2 |\Delta U|_{Js}^2, \\
|\Delta U|^2{}_{Js} &= 4k_B T_s R_s E_J,
\end{aligned}
$$

and the $1/f$ or flicker noise by

$$
\begin{aligned}
|\Delta I_f(w)|_{flicker}^2 &= \left|s_{21}^{int}(w)\right|^2 |\Delta P_e|_{flicker}^2, \\
|\Delta P_e(w)|_{flicker}^2 &= \frac{\left(\frac{\Delta R_{f,flicker}}{R_{f0}}\right)^2 R_{f0}^2}{w^\alpha} I_{f0}^2,
\end{aligned}
$$

where $\left(\frac{\Delta R_{f,flicker}}{R_{f0}}\right)$ and α are parameters to be extracted from the data.

Additionally, the SQUID produces an amount i_{sq} of white noise that is directly added to the SQUID output current, decoupled from feedbacks in the readout circuit, and peaks in the spectrum from electromagnetic interference are added with height $p_0/p_1/p_2$ at the frequencies 50/150/250 Hz, and scaled to the observed heights in the data as well.

The noise contributions are assumed to be Gaussian and independent with respect to each other and can therefore be summed quadratically.

App. B: Parameters used in the simulation

We report in Tab. 1 the parameters used for the simulation that was introduced in Sec. 2.

We calculated the heat capacities for the absorbers following the Debye model, with the Debye temperature of lithium aluminate calculated from the elasticity constants from Ref. [29], with the value 429 K, and for sapphire taken from Ref. [30], with the value 1041 K. We evaluate them at the transition temperature T_c and scaled them to the absorber volume V_a.

We calculated the heat capacities for the tungsten TES with the Sommerfeld constants taken from Ref. [31]. The values are evaluated at T_c and scaled to the TES volume V_f. The stated value is without the increase by factor 2.43 appearing in the transition curve, which is dynamically calculated when the differential equations are numerically solved.

The calculated value does only include the part of the tungsten film that is not covered by aluminum. The aluminum-covered part functions solely as an athermal phonon collector.

The normal conducting resistance of the TES R_{f0} and its transition temperature T_c are measured values.

R_s, η, L, i_{sq}, I_H, IB_{min} and IB_{max} are known values of the setup. η is the conversion factor that translates a current in the SQUID branch of the readout circuit to the observable voltage, determined by the SQUID settings.

The thermal couplings, τ_n, and ε were fitted to the pulse shape parameters of observed absorber recoils.

δ, δ_H, R_H, $\frac{\Delta R_{f,flicker}}{R_{f0}}$, k, β, E_J, p_0, p_1, p_2 and α are adjusted to match the measured data. The parameter k controls the steepness of the TES transition curve, which we simplify as a logistics function. The derivative of the transition curve at its steepest point has the value $4kR_{f0}$. The parameter β is a scale factor between the injected test pulses and the constant heating current. The values p_0, p_1, and p_2 are coefficients of numerical templates of the electrical poles in the frequency spectrum.

All stated values are effective parameters that, within our model, reproduce the behavior of the real-world detectors. They are not necessarily a unique combination that reproduces the behavior of the real-world detectors and do not

Quantity	Li1P	Li1L	Li2P
V_f (mm^3)	$4.08 \cdot 10^{-4}$	$2.04 \cdot 10^{-4}$	$4.08 \cdot 10^{-4}$
V_a (mm^3)	$4 \cdot 10^3$	$2 \cdot 10^2$	$4 \cdot 10^3$
C_e (pJ/mK)	$2.11{\cdot}10^{-3}$	$9.75{\cdot}10^{-4}$	$2.5{\cdot}10^{-3}$
C_a (pJ/mK)	0.113	$1.61{\cdot}10^{-4}$	$9.7{\cdot}10^{-2}$
G_{eb} (pW/mK)	0.15	0.5	0.138
G_{ab} (pW/mK)	1.5	$5{\cdot}10^{-3}$	1.16
G_{ea} (pW/mK)	$9.72{\cdot}10^{-2}$	$5{\cdot}10^{-3}$	$9.45{\cdot}10^{-2}$
τ_n (s)	$3.82{\cdot}10^{-4}$	$9.4{\cdot}10^{-5}$	$4{\cdot}10^{-4}$
τ_{TP} (s)	$4.97{\cdot}10^{-3}$	$2.98{\cdot}10^{-3}$	$4.97{\cdot}10^{-3}$
ε	0.132	0.37	0.104
δ	0.047	0.587	0.129
δ_H	0.05	0.99	0.2
R_s (Ω)	$4{\cdot}10^{-2}$	$4{\cdot}10^{-2}$	$4{\cdot}10^{-2}$
R_H (Ω)	9.16	7.06	1.07
R_{f0} (Ω)	0.11	0.115	0.1
L (H)	$3.5{\cdot}10^{-7}$	$3.5{\cdot}10^{-7}$	$3.5{\cdot}10^{-7}$
T_c (mK)	30.7	23	29.4
k (1/mK)	4.4	13.5	5.52
β	$2.25{\cdot}10^{-2}$	$6.25{\cdot}10^{-3}$	$2{\cdot}10^{-2}$
I_H (μA)	4.8	0.904	8.27
i_{sq} (pA/$\sqrt{\text{Hz}}$)	1.2	1.2	1.2
E_J	6	4	7
$\frac{\Delta R_{f,flicker}}{R_{f0}}$ (pJ)	$3.5{\cdot}10^{-4}$	$9{\cdot}10^{-4}$	$3{\cdot}10^{-5}$
α	2.5	1.5	1.5
p_0	$1.5 \cdot 10^{-5}$	$3 \cdot 10^{-5}$	$3 \cdot 10^{-5}$
p_1	$1 \cdot 10^{-5}$	$2 \cdot 10^{-5}$	$2 \cdot 10^{-5}$
p_2	$1.5 \cdot 10^{-5}$	$2 \cdot 10^{-5}$	$2 \cdot 10^{-5}$
IB_{min} (μA)	0.5	0.5	0.5
IB_{max} (μA)	17.9	17.9	17.9
η (V/μA)	5.77	5.77	5.77

Table 1: Values used in the simulation of the detectors Li1P, Li1L and Li2P. See text, Sec. 2 and App. A for definitions.

necessarily correspond exactly to the true underlying physical quantities, as several effects, e.g. that of the spatial resolution, are absorbed in the parameters. For this reason, we do not report uncertainties, but only the exact values we used.

When we randomized the values stated in Tab. 1 for the detector version used for training in Sec. 4 we multiplied them by a Gaussian random number $\sim \mathcal{N}(\mu = 1, \sigma = 0.2)$. We tested for every detector version if the transition is reachable with the available heating and bias currents and resampled the parameters with a new random seed otherwise.

When we adjusted the values to two TES versions used in Sec. 4.1, we copied the parameters of the TES to a third thermal component, with an independent readout circuit, but divided the collection efficiency ε by two, assuming that the athermal phonon population would likely distribute among the two TES.

App. C: Details of models and training

We use for the training and evaluation of models the Vienna CLIP computing cluster. Neural networks were implemented with the PyTorch library [32].

App. C.1: Details of training in virtual environment

We run individual single-CPU jobs for the training of each detector version, they take between one and two hours to be completed. For the detector versions with two TES modules, we require between 6 and 8 hours of training time. Depending on the number of gradient steps about half of the time is spent in the simulation which is in our current implementation only CPU-supported. We would therefore not gain significantly through operation on a GPU.

The policy and value functions of our SAC agents are 2-layer neural networks with 256 nodes in each layer, and ReLU activation functions.

We optimize their weights with the ADAM optimizer [33], using a weights decay of $1 \cdot 10^{-5}$. The learning rates, batch sizes, γ values of the temporal difference method, and the number of gradient steps after each environment step are different in each phase of detector versions and stated in Fig. 5. For training the 2 TES detector versions, we used the same hyperparameters as in the first phase. We put the τ update parameter of the SAC algorithm to the value 0.005 and the initial entropy coefficient α to 0.2. Training is started as soon as one full batch of state transitions is collected in the replay buffer. Gradients are clipped at the norm 0.5.

App. C.2: Details of live training

For live training on the CRESST setup, we connected a computing node of the Vienna CLIP cluster with an SSH tunnel to the MQTT broker that was operated in the LNGS network.

We used for all performance runs the same neural network architectures for policy and value functions as for training in the virtual environment. The neural networks were trained with the ADAM optimizer, with a batch size of 16 and 20 gradient steps after each test pulse, a learning rate of $3 \cdot 10^{-4}$ and weight decay of $1 \cdot 10^{-5}$. Usually performing the 20 gradient steps took much less time than the interval between test pulses, the total time required for training was therefore determined by the measurement time on the experiment. Gradients were clipped at the norm 0.5.

We set the initial entropy of the SAC algorithm to 0.2, and the τ update parameter to $5 \cdot 10^{-3}$. Several hyperparameters and settings of the RL problem were varied throughout the six performance runs, and an overview of them is contained in Tab. 2.

Detector run	**Li1P 1**	**Li1L 1**	**Li2P 1**	**Li2P 2**	**Li2P 3**	**Li1L 2**
γ	0.9	0.99	0.99	0.99	0.99	0.99
Reward	- Eq. (6)	- Eq. (6)	- Eq. (6)	- Eq. (6)	Eq. (7)	Eq. (7)
TP int. (s)	20	20	10	10	10	10
DAC_{max}	10	10	5	5	5	10
IB_{min}	0.5	0.1	0.5	0.5	0.5	0.1
IB_{max}	5	3	5	5	5	3
ADC range	±10	±0.3	±1	±1	±1	±0.3
TPA in state	yes	no	no	no	no	yes
CPH in state	no	no	no	no	yes	yes
ADCs/IB	no	no	yes	no	yes	yes

Table 2: Hyperparameter and settings of the RL problem used for the six performance runs on the CRESST underground setup. See the text for explanations.

The reward function was varied between using the weighting with inverse TPA values or not (see Sec. 3). The time interval between test pulses was for two runs set to a higher value than the default of 10 seconds, to test if thermal relaxations on larger time scales have an impact on the training. The normalization intervals for the DAC and IB values were individually adjusted for the detectors, as well as the value range of the analog-digital converter (ADC). The state space of the RL problem was adjusted to contain the TPA and CPH values for a subset of the runs. Furthermore, the division of the values that scale with the ADC and IB, PH, and RMS, by the IB value is done for a subset of the runs.

App. C.3: Details of Transformer training

We use for our experiments the Huggingface DT implementation [34]. We test three architectures of different sizes: the architecture from the original Ref. [11] with 1 M parameters, which was shown to perform well on individual RL environments; and the architectures with 10 M and 40 M parameters from Ref. [12] that were shown to perform well in multi-task settings. Their architectures and the hyperparameters used during training are contained in Tab. 3. Parameters not mentioned here are the Huggingface defaults. We use a learning rate scheduler that increases the learning rate to its nominal value linearly in the first 10 % of targeted gradient steps and decreases it afterward linearly to zero until the target number of training epochs is reached. The context length was for all architectures set to 20 environment steps.

During training, we evaluate the model performance after 500 optimizer steps on the validation set. The resulting validation losses are contained in Fig. 12. Discontinuities in the curves are points where we restarted training from recorded checkpoints but with different states of the learning rate scheduler. For reference we also plot a curve of the smallest architecture, trained and evaluated only on only one detector version, where it reaches identical performance as the SAC agent, to obtain a benchmark for an ideal behavior.

The training was done on a computing node with a single Tesla V100 GPU.

References

1. A. H. Abdelhameed et al. First results from the CRESST-III low-mass dark matter program. *Phys. Rev. D*, 100:102002, Nov 2019.
2. C. Collaboration et al. Results on sub-GeV Dark Matter from a 10 eV Threshold CRESST-III Silicon Detector, 2022.
3. J. Billard et al. Direct detection of dark matter—APPEC committee report*. *Reports on Progress in Physics*, 85(5):056201, apr 2022.
4. R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT press, 2018.
5. J. St. John et al. Real-time artificial intelligence for accelerator control: A study at the Fermilab Booster. *Phys. Rev. Accel. Beams*, 24:104601, Oct 2021.
6. F. M. Velotti et al. Towards automatic setup of 18 MeV electron beamline using machine learning. *Machine Learning: Science and Technology*, 4(2):025016, apr 2023.
7. J. Degrave et al. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602(7897):414–419, February 2022.
8. H. P. Nautrup et al. Optimizing Quantum Error Correction Codes with Reinforcement Learning. *Quantum*, 3:215, dec 2019.
9. T. Haarnoja et al. Soft Actor-Critic Algorithms and Applications. *ArXiv*, 1812.05905, 2018.
10. G. Angloher et al. Testing spin-dependent dark matter interactions with lithium aluminate targets in CRESST-III. *Phys. Rev. D*, 106:092008, Nov 2022.
11. L. Chen et al. Decision Transformer: Reinforcement Learning via Sequence Modeling. *ArXiv*, 2106.01345, 2021.

Parameters	**1 M**	**10 M**	**40 M**
Number of layers	3	4	6
Number of attention heads	1	8	12
Embedding dimension	128	512	768
Batch size	64	64	256
Dropout	0.1	0.1	0.1
Learning rate	10^{-4}	10^{-4}	$3 \cdot 10^{-4}$
Grad norm clip	0.25	0.25	1.
Weight decay	10^{-4}	10^{-4}	0.
MSE test loss	0.11	$1.14 \cdot 10^{-2}$	$5.6 \cdot 10^{-3}$

Table 3: Hyperparameters and test loss values of the used Transformer architectures.

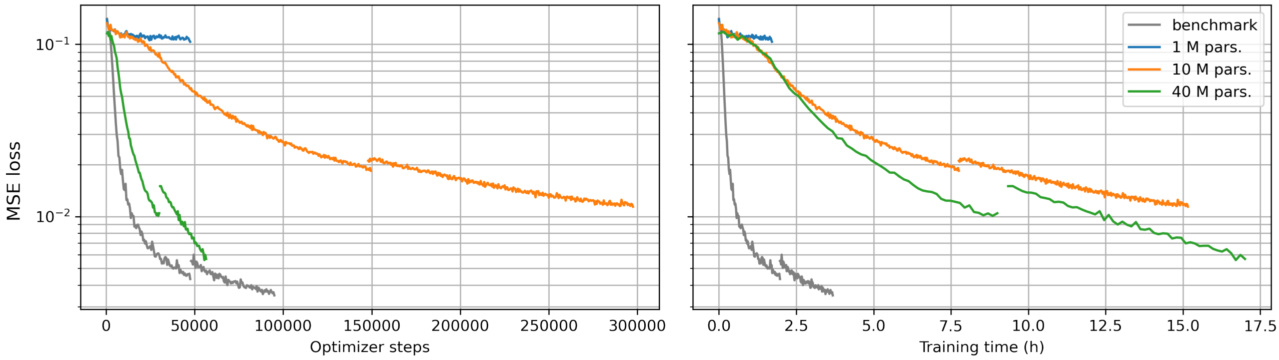

Fig. 12: Evaluation loss taken during training of the Transformer model with 1 M (blue) / 10 M (orange) / 40 M (green) parameters, depending on the number of gradient steps (left) and the elapsed training time (right). As benchmark (grey) the evaluation loss of the 1 M model is shown when only trained on one detector version, which reaches a similar performance as the SAC agent on this detector.

12. K.-H. Lee et al. Multi-Game Decision Transformers. *ArXiv*, 2205.15241, 2022.
13. F. Pröbst et al. Model for cryogenic particle detectors with superconducting phase transition thermometers. *Journal of Low Temperature Physics*, 100(1):69–104, July 1995.
14. K. Irwin and G. Hilton. Transition-Edge Sensors. In C. Enss, editor, *Cryogenic Particle Detection*, pp. 63–150. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.
15. M. Tinkham. *Introduction to Superconductivity*. Dover Publications, 2 edition, June 2004.
16. M. Galeazzi and D. McCammon. Microcalorimeter and bolometer model. *Journal of Applied Physics*, 93(8):4856–4869, March 2003. _eprint: https://pubs.aip.org/aip/jap/article-pdf/93/8/4856/10626979/4856_1_online.pdf.
17. P. Virtanen et al. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.
18. M. Carrettoni and O. Cremonesi. Generation of noise time series with arbitrary power spectrum. *Computer Physics Communications*, 181(12):1982–1985, 2010.
19. G. Angloher et al. Results on light dark matter particles with a low-threshold CRESST-II detector. *The European Physical Journal C*, 76(1):25, January 2016.
20. G. Angloher et al. First measurements of remoTES cryogenic calorimeters: Easy-to-fabricate particle detectors for a wide choice of target materials. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 1045:167532, January 2023.
21. Y. Chebotar et al. Closing the Sim-to-Real Loop: Adapting Simulation Randomization with Real World Experience, 2019.
22. A. A. Rusu et al. Sim-to-Real Robot Learning from Pixels with Progressive Nets, 2018.
23. J. Tobin et al. Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World, 2017.
24. A. Vaswani et al. Attention is All you Need. In I. Guyon et al., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.
25. A. Radford et al. Improving language understanding by generative pre-training. 2018.
26. S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

27. G. Angloher et al. Towards an automated data cleaning with deep learning in CRESST. *The European Physical Journal Plus*, 138(1):100, January 2023.
28. F. Wagner et al. Cait: Analysis Toolkit for Cryogenic Particle Detectors in Python. *Computing and Software for Big Science*, 6(1):19, December 2022.
29. A. Jain et al. Commentary: The Materials Project: A materials genome approach to accelerating materials innovation. *APL Materials*, 1(1), July 2013. _eprint: https://pubs.aip.org/aip/apm/article-pdf/doi/10.1063/1.4812323/13163869/011002_1_online.pdf.
30. E. Pantić. *Performance of Cryogenic Light detectors in the CRESST-II Dark Matter Search*. PhD thesis, TU Munich, 2008.
31. C. Kittel. *Introduction to Solid State Physics*. Wiley, 8 edition, 2004.
32. A. Paszke et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019.
33. D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization, 2017.
34. T. Wolf et al. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, Online, October 2020. Association for Computational Linguistics.