ACTION-BED: Task-Driven Bayesian Experimental Design with Singly Intractable Objectives

Tom Rossa
Department of Statistics
University of Oxford
tom.rossa@stats.ox.ac.uk

Angus Phillips
Department of Statistics
University of Oxford
angus.phillips@stats.ox.ac.uk

Tom Rainforth
Department of Statistics
University of Oxford
rainforth@stats.ox.ac.uk

Abstract

Bayesian experimental design (BED) has traditionally been based on maximising expected uncertainty reductions from prior to posterior. A major shortfall of this approach is that it leads to doubly intractable objectives that are difficult to optimise, while customising them to particular downstream tasks of interest can also be difficult. Following first principles decision theory, we demonstrate that BED can alternatively be formulated in terms of an expected future loss (EFL) on downstream actions, providing a simple and naturally task-driven framework. Critically, we then show that all such EFLs can be rearranged into *singly intractable objectives* that can be jointly optimised with respect to both the design policy and a downstream *action policy* using stochastic gradients, an approach we refer to as ACTION-BED. This formulation further sidesteps the need for any explicit posterior or marginal likelihood estimation and is naturally implicit, requiring only the ability to sample from the joint model over model parameters and data, and evaluate the downstream loss function. It thus allows design policies to be learned more effectively, efficiently, and simply than existing methods, while providing easy customisation to different downstream tasks and losses.

1 Introduction

Bayesian Experimental Design (BED) [14, 19, 32, 52, 53, 64] is a principled framework for the optimal acquisition of data, in which designs are chosen to maximise the expected uncertainty reduction (EUR) from prior to posterior over unknown values of interest. Typically the uncertainty measure is taken to be the Shannon entropy [67], leading to the classical expected information gain (EIG) objective [52]. In this perspective, data is valued according to how it changes our belief distribution over unknown values, which is theoretically justified by the fact that downstream Bayes-optimal actions only depend on observed data through the updated belief distribution [65].

However, there are two major issues with using this framework in practice. First, defining uncertainties that are customised to particular downstream tasks of interest can be challenging [44], noting that the data gathered is rarely the end goal in itself and is instead used to aid downstream estimation, prediction, or decision-making. In particular, common metrics like Shannon entropy constitute generic uncertainties which fail to reflect that not all information is equally useful for particular tasks of interest.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

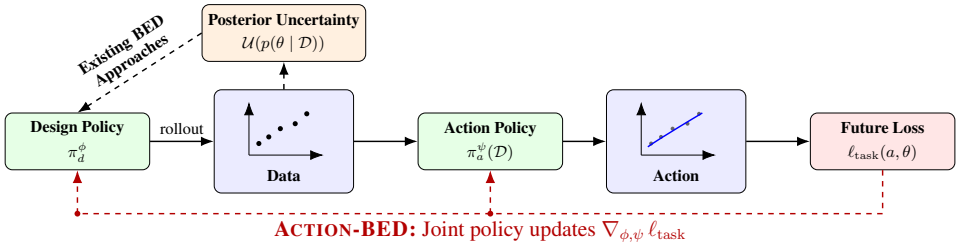


Figure 1: **ACTION-BED**; our task-driven, singly intractable algorithm for policy-based BED. Rather than targeting uncertainty reductions, ACTION-BED jointly optimises a downstream action policy alongside the design policy to minimise the expected future loss using stochastic gradient descent.

Second, EURs are generally doubly intractable [64]: without an analytic posterior, the EUR is inevitably a nested expectation [63] as meaningful uncertainty measures are nonlinear [18, 27, 54, 66]. This makes estimating and optimising EURs very challenging as standard estimators cannot be applied, while the specialised computational strategies proposed to deal with EURs [2, 23, 24, 28, 39–41] are often only applicable to EIGs and tend to suffer from high computational costs and bias. For example, one can look to circumvent the double intractability of EURs by learning amortised posterior approximations [4, 25, 61], often learning this alongside the design/policy optimisation using variational formulations [12, 26, 35–37, 68]. However, this amortised posterior learning is still itself a very challenging task and inaccuracies in the posterior approximations can undermine the resulting data acquisition.

We propose to address these challenges with a fundamental move away from the EUR framework by instead defining the utility of data through *the performance of the downstream actions it enables*. Specifically, using first principles Bayesian decision theory, we show that the Bayes-optimal data gathering policy can be formulated as the minimiser of the *expected future loss* (EFL) of downstream Bayes-optimal actions, allowing for the definition of experimental design objectives that are naturally customised to our downstream tasks and losses of interest.

While the set of unique objectives indexed by the EFL and EUR frameworks turns out to be equivalent, our critical insight is that the former allows for a simple rearrangement into a *singly intractable* objective that we jointly optimise with respect to a design policy and a *downstream action policy*. This results in a much simpler and computationally tractable policy-based framework than existing approaches that we call **ACTION-BED**. As shown in Figure 1, ACTION-BED avoids the need to directly perform posterior approximations or explicitly estimate any uncertainties: it simply requires the minimisation of an expected loss with respect to the joint policy parameters, which can be done using standard stochastic gradient descent methods. Furthermore, the task-driven EFL objectives it uses are *naturally implicit*, requiring only reparametrisable simulations from the joint model over parameters and data, and a differentiable loss function. These benefits can substantially simplify the policy training and we empirically find that ACTION-BED often significantly outperforms previous state-of-the-art in terms of EIG values even when targeting different downstream tasks and losses.

In summary, we propose a paradigm shift in BED from thinking about uncertainties to future losses. Not only does this allow us to easily customise design policy learning to downstream tasks of interest, it also provides substantial practical benefits via our ACTION-BED framework, which circumvents the double intractability of EURs, avoids explicit posterior approximation or model density evaluations, and ultimately collects more informative and targeted data than existing approaches. A reference implementation of ActionBED is provided in the TestBED PyTorch library (`http://github.com/testbed-benchmark/testbed`); its accompanying paper also provides additional empirical evaluations of ACTION-BED that further confirm its efficacy compared with baseline approaches [43].

2 Background

Bayesian experimental design. Bayesian experimental design (BED) [14, 17, 19, 22, 32, 52, 53, 64] is typically defined as choosing designs to maximise the reduction in a particular uncertainty measure between the prior and the posterior. Namely, let $p(\theta)p(y|\theta,\xi)$ denote the joint generating process over outcome $y \in \mathcal{Y}$ and variables of interest $\theta \in \Theta$ given experimental design $\xi \in \Xi$. Then the expected uncertainty reduction (EUR) in θ for an uncertainty measure $\mathcal{U}$ and design ξ is:

$$\text{EUR}_\theta(\xi) = \mathbb{E}_{p(\theta)p(y|\theta,\xi)}\left[\mathcal{U}[p(\theta)] - \mathcal{U}[p(\theta|y,\xi)]\right]. \tag{1}$$

Typically, $\mathcal{U}$ is taken to be the Shannon entropy, $\mathcal{U}[q(\theta)] = \mathbb{E}_{q(\theta)}[-\log q(\theta)]$, in which case the EUR corresponds to the *expected information gain* (EIG) in θ, i.e. the mutual information between θ and y.

Sequential and policy-based BED. BED is often most impactful in sequential settings where one performs an experiment adaptively, say over T rounds. At each round t, a design $\xi_t \in \Xi_t$ is proposed and outcome $y_t \in \mathcal{Y}_t$ is observed according to our proposed model $p(y_t|\xi_t, \theta)$. Let $h_t := \{(\xi_1, y_1), \dots, (\xi_t, y_t)\}$ be the experimental history to time t, referred to in generality as the data. Traditional approaches to sequential BED [57] would at each step infer the posterior and select the next design by optimising the EIG, however this process is prohibitively expensive. Instead, policy-based approaches [10, 12, 20, 24, 30, 35–37, 41, 42, 61, 68, 69] train an adaptive policy network $\pi_d \in \Pi_d$ which maps historical data h_t onto optimal next designs ξ_{t+1}. The policy is generally trained to maximise (an approximation of) the overall EUR from observing the full experimental history. This results in an acquisition policy which acts non-myopically to maximise the EUR from observing the full dataset and which can be deployed almost instantaneously. Note that the optimal design policy is always deterministic [53] and so we will assume deterministic policies throughout, but our methods can equally be applied with probabilistic design policies if desired.

Optimising EIG. The longstanding challenge with EUR objectives, such as the EIG, is their fundamental double intractability. This arises because the uncertainty measures considered are non-linear functionals of the intractable posterior distribution [18, 27, 54, 66], which are then nested within expectations over the data generating process. Much work has been devoted to developing estimation and optimisation procedures for these objectives, in particular for the EIG [2, 15, 24–26, 28, 29, 33, 34, 39–42, 46, 47, 63]. Of particular relevance is the family of variational bounds that allow the EIG to be represented as an optimisation over some kind of amortised posterior approximation [25, 26] (which is sometimes indirectly defined through an approximation of the marginal likelihood or its score [61]). For example, the Barber Agakov (BA) bound is given by [4]:

$$\mathcal{I}_T(\pi_d) \geq \mathcal{L}(\pi_d, \psi) = \mathbb{E}_{p(\theta)p(y_{1:T}|\theta;\pi_d)}\left[\log q_\psi(\theta|h_T) - \log p(\theta)\right] \tag{2}$$

where $q_\psi(\theta|h_T)$ is an amortised approximation of the posterior distribution with variational parameters $\psi \in \Psi$. If there exists some $\psi \in \Psi$ such that $q_\psi(\theta|h_T)$ perfectly captures the true posterior $p(\theta|h_T)$ for all possible h_T, then $\max_{\psi\in\Psi} \mathcal{L}(\pi_d, \psi) = \mathcal{I}_T(\pi_d)$. This bound then allows a singly intractable joint optimisation of both design policy π_d and variational posterior q_ψ. However, in practice, it is limited by the variational family and difficulty of learning accurate amortised posteriors.

3 Loss-Driven Bayesian Experimental Design

As we have established, the de-facto perspective on BED as maximising EURs leads to objectives which are fundamentally doubly intractable and therefore pose a real computational barrier to their adoption, particularly at scale. Furthermore, variational formulations for circumventing this rely on learning accurate amortised posterior approximations, which is typically challenging and often unnecessary when the posterior itself is not the downstream goal of the experiment. They are also specific to the EIG, and there is a lack of more general computationally viable frameworks for other uncertainty measures, in particular customising to downstream tasks and losses of interest.

Instead we propose an entirely loss-driven approach, inspired by the principle that the value of data should be measured by how well it allows us to complete downstream tasks. Grounding ourselves in first principles Bayesian decision theory (BDT), we show Bayes-optimal design policies take the form of the minimisers of an expected future loss when future actions are made Bayes-optimally. We then show how this can be turned into a practical, computationally tractable BED algorithm, ACTION-BED, which solves the aforementioned challenges and is additionally loss-driven and implicit, only relying on joint simulation of parameters and data under the proposed model. It further has the benefit of avoiding the need for full posterior approximation, even after the data has been gathered.

3.1 Bayesian Experimental Design from First Principles

Bayesian decision theory [7, 16, 19, 53, 62, 65, 71, 72] dictates that a rational agent should plan ahead in a way that assumes that they will act optimally given the information available to them in the future. We can thus derive a Bayes-optimal design policy from first principles, by starting with terminal actions taken after all data is gathered and then working backwards through our sequential decisions.

To derive the optimal downstream action given acquired data, we note that the Savage axioms imply the existence, for any rational agent, of a loss function on actions and the state of the world [65],

$\ell : \mathcal{A} \times \Theta \to \mathbb{R}$, which encapsulates our preferences over the consequences from the actions we take, and a subjective belief model on the world given observed data $p(\theta|h_T)$. Here $\theta \in \Theta$ now denotes whatever aspect of the true world state directly influences our loss function. The Bayes-optimal action, a^B, is then the minimiser of the Bayesian expected loss, $a^B = \arg\min_{a \in \mathcal{A}} \mathbb{E}_{p(\theta|h_T)} [\ell(a, \theta)]$. This is easily generalised to cases where the loss depends on the data h_T itself, but we omit this dependence for ease of notation (see Appendix A).

For a given belief system, loss function, and action space, the minimal achievable Bayesian expected loss, $\min_{a \in \mathcal{A}} \mathbb{E}_{p(\theta|h_T)} [\ell(a, \theta)]$, is now only a function of the data we observe. It is analogous to a value function in reinforcement learning (RL) [56]: better data will provide a more certain belief state and, in turn, a better achievable expected downstream loss. This downstream value of our data is independent of the policy that was used to gather it [61], with the policy instead influencing the belief distribution on data that might be generated, $p(h_T; \pi_d)$. The existence of this belief distribution is again implied for a rational agent by the Savage axioms, noting that we are uncertain about both h_T and θ before the experiment. Further, our beliefs on θ should not depend on our data gathering policy, so we should have that our "prior" $p(\theta) = \int p(h_T; \pi_d) p(\theta|h_T) dh_T$ is independent of π_d.

As minimising the loss is our ultimate objective and a rational agent will act optimally once the data is gathered, the Bayes-optimal data gathering policy is then the minimiser of the *expected future loss* (EFL) when downstream actions are themselves made Bayes-optimally as follows.

Definition 1 (Bayes-Optimal Experimental Design Policy)**.** *The Bayes-optimal sequential experimental design policy π_d^B for a given belief model and downstream loss pairing is defined as:*

$$\pi_d^B = \arg\min_{\pi_d \in \Pi_d} \mathrm{EFL}_\ell^B(\pi_d) \quad \text{where} \quad \mathrm{EFL}_\ell^B(\pi_d) = \mathbb{E}_{p(h_T; \pi_d)} \left[\min_{a \in \mathcal{A}} \mathbb{E}_{p(\theta|h_T)} [\ell(a, \theta)] \right]. \tag{3}$$

We note that similar theoretical EFL formulations of the Bayes-optimal experimental designs have already been established in the classical BED literature, most notably Lindley [53] and Dawid [17]. However, they only considered individual design decisions rather than design *policies* and, moreover, it has not since been directly used for deriving computationally viable BED approaches. Instead, the BED field has since almost exclusively relied on the mathematical equivalence of this EFL formulation to EURs (see Section 3.3) when constructing practical methods.

3.2 Singly Intractable and Loss-Driven BED via Joint Policy Optimisation

While Definition 1 provides a precise formulation of the Bayes-optimal data gathering policy as the minimiser of a task-driven EFL, directly solving Equation (3) is very difficult as it involves a nested optimisation due to the dependence of the optimal action a^B on the observed data h_T. Our key insight is that we can switch from optimising individual actions to optimising an *action policy*, π_a, which maps from observed data to downstream actions. As we show below, this leads to a singly intractable joint objective for design and action policy training which is compatible with any downstream loss.

Let $\pi_a \in \Pi_a := \{\pi \mid \pi : \mathcal{H}_T \to \mathcal{A}\}$, where $\mathcal{H}_T = \prod_{t=1}^T (\mathcal{Y}_t \times \Xi_t)$, be an *action policy* mapping observed data to downstream actions. Noting that Π_a contains all admissible measurable mappings from the data to actions and the optimal action is defined on a data-by-data basis for a given model and loss, under standard measurability assumptions detailed in Appendix A, we can lift the optimisation over individual actions in the EFL outside the expectation to become an optimisation over action policies as follows:

$$\mathrm{EFL}_\ell^B(\pi_d) = \min_{\pi_a \in \Pi_a} \mathbb{E}_{p(h_T; \pi_d) p(\theta|h_T)} [\ell(\pi_a(h_T), \theta)]. \tag{4}$$

While the minimisation over action policies in Equation (4) does not usually have an analytic solution, we can instead think of the EFL under more general downstream action policies before jointly optimising for both π_a and π_d, which will still recover the Bayes-optimal policies for both actions:

$$\begin{aligned} (\pi_a, \pi_d)^B &= \arg\min_{(\pi_a, \pi_d) \in \Pi_a \times \Pi_d} \mathrm{EFL}_\ell(\pi_d, \pi_a), \\ \text{where} \quad \mathrm{EFL}_\ell(\pi_d, \pi_a) &= \mathbb{E}_{p(\theta) p(y_{1:T}|\theta; \pi_d)} [\ell(\pi_a(h_T), \theta)], \end{aligned} \tag{5}$$

and we have rewritten our belief model using $p(\theta) p(y_{1:T}|\theta; \pi_d) = p(h_T; \pi_d) p(\theta|h_T)$.

We refer to approaches that directly train policies using Equation (5) as **ACTION-BED**. Its seemingly straightforward reinterpretation of experimental design as a joint optimisation problem now

has remarkably impactful consequences. Indeed the EFL represents a completely general, singly intractable, naturally implicit and loss-driven objective for optimal experimental design. It never requires the approximation of posterior distributions or evaluation of unknown densities, it simply relies on simulations from the model. As we will see in Section 4, parameterising both action and design policies as deep neural networks leads to a practical end-to-end experimental design algorithm.

3.3 Recovering the Expected Uncertainty Reduction Formulation

Traditional EUR objectives can be formalised and reconciled with our BED formulation through the framework of proper scoring rules. A scoring rule is a function $s : \mathcal{P}_\theta \times \Theta \to \mathbb{R}$ which assigns a score $s(p_\theta, \theta)$ to a predictive distribution $p_\theta \in \mathcal{P}_\theta$ when the realisation of a random variable Θ is θ. The scoring rule is (strictly) proper if its expectation over Θ is (strictly) minimised when the predicted distribution is exactly the distribution of Θ. Following ideas introduced by [9, 17, 19], we can define general uncertainty measures, known as *generalised entropies*, using $h_s[q(\theta)] = \mathbb{E}_{q(\theta)}[s(q(\cdot), \theta)]$. We then define a classical BED objective via the expected posterior uncertainty (EPU):

$$\text{EPU}_s(\pi_d) = \mathbb{E}_{p(y_{1:T};\pi_d)}\left[h_s[p(\theta|h_T)]\right] = \mathbb{E}_{p(\theta)p(y_{1:T}|\theta;\pi_d)}\left[s(p_\theta(\cdot|h_T), \theta)\right]. \tag{6}$$

Assuming the scoring rule does not directly depend on π_d or $y_{1:T}$, the uncertainty of the prior $h_s[p(\theta)]$ is constant in π_d, so minimising such an EPU is equivalent to maximising the more commonly seen EUR given by $\text{EUR}_\theta(\pi_d) = h_s[p(\theta)] - \text{EPU}_s(\pi_d)$, where our uncertainty measure $\mathcal{U}$ is h_s.

We now show that via an equivalence between loss-action pairings and proper scoring rules, any optimal BED policy according to Definition 1 can be represented as the minimiser of an EPU for some scoring rule s. Conversely, minimising the EPU associated with any proper scoring rule yields a Bayes-optimal design policy for at least one belief-loss pairing (see Appendix B.1 for proof).

Theorem 1 (Equivalence of EFL & EPU)**.** *Let $s : \mathcal{P}_\theta \times \Theta \to \bar{\mathbb{R}}$ be a proper scoring rule on Θ. Then*

$$\pi_d^B \in \arg\min_{\pi_d \in \Pi_d} \mathbb{E}_{p(\theta)p(y_{1:T}|\theta;\pi_d)}\left[s\big(p_\theta(\cdot|h_T), \theta\big)\right] \tag{7}$$

is a Bayes-optimal design policy for the belief model $p(\theta)p(y_{1:T}|\theta;\pi_d)$ and for any downstream loss $\ell \in \mathcal{L}_s$ satisfying the regularity conditions stated in Appendix B.1, where

$$\mathcal{L}_s = \left\{\ell : \forall q_\theta \in \mathcal{P}_\theta,\ \inf_{a \in \mathcal{A}} \mathbb{E}_{\theta \sim q_\theta}\left[\ell(a, \theta)\right] = \mathbb{E}_{\theta \sim q_\theta}\left[s(q_\theta, \theta)\right]\right\}. \tag{8}$$

Furthermore, the set $\mathcal{L}_s$ is never empty, and any admissible loss function ℓ induces one or more corresponding proper scoring rules, though not necessarily strictly proper.

Consequently, minimising EFL is equivalent to minimising EPU under the conditions of Theorem 1, and in turn maximising EUR provided the scoring rule does not depend on π_d or $y_{1:T}$ (see Appendix B.1 for discussions of the case where this does not hold). For example, maximising EIG is equivalent to minimising EPU with the log score $s(q_\theta(\cdot), \theta) = -\log q_\theta(\theta)$, for which the canonical downstream loss is the log loss and the optimal action is the Bayesian posterior itself.

Theorem 1 thus reconciles expected future loss and future uncertainty as different representations of the same underlying decision problem. The key distinction is thus whether we specify the problem by directly specifying uncertainty measures or an explicit downstream task and loss. The latter makes it easier to tailor objectives for specific downstream tasks, such as targeting the point estimation of some quantity of interest, or incorporating in real-world decision-making, such as business decisions or choosing medical treatments. It also makes clear how choices of uncertainty measures in the EUR framework implicitly target particular downstream tasks and losses, e.g. EIG targets log loss on probabilistic predictions, expected variance reduction targets point estimation with squared error loss, etc.

Critically, though, there are also significant practical distinctions between the viewpoints in terms of their computational viability, with the EFL formulation leading to the general purpose singly intractable problem formulation we introduced in Equation (5). Interestingly, this formulation recovers existing variational formulations [4, 25] as a special case when using an EFL that is equivalent to the EIG (as here the optimal action is the posterior, see Appendix B.2). However, it crucially generalises to other action spaces and losses, and it turns out the action policies we need to learn for other downstream losses are often much simpler than those required for the EIG. In particular, when the required downstream actions correspond to point estimates or explicit decisions rather than distributions, then the action policy is a simple functional mapping to single values, avoiding the

need to implicitly estimate or approximate the posterior. For example, if using squared error loss on an estimation task, then the action policy only needs to learn to map from data to the posterior mean. This can substantially simplify the policy training as we observe in our experiments (see Section 6).

4 ACTION-BED

We now show how our ACTION-BED framework, based on the joint optimisation of the EFL with respect to action and design policies as per Equation (5), can be realised as a practical end-to-end algorithm for task-driven BED. Specifically, building on the deep adaptive design (DAD) approach of [24, 42], we propose to directly parameterise both policies using deep neural networks, denoted π_d^ϕ and π_a^ψ, with parameters $\phi \in \Phi$ and $\psi \in \Psi$. We then train these networks using end-to-end stochastic gradient descent on the expected future loss for rollouts drawn from the model, as shown in Algorithm 1. Empirically, it is often helpful to initialise $\psi \in \Psi$ by pre-training using a fixed random design policy (Appendix E.2), following which our algorithm remains unchanged (with ψ_0 set to this learned initialisation).

Require: Policies π_d^ϕ, π_a^ψ, initialisations ϕ_0, ψ_0
for $k = 0, 1, \dots, K-1$ **do**
  $\theta^{1:B}, \varepsilon_{1:T}^{1:B} \sim p(\theta)q(\varepsilon_{1:T})$
  $g_k^\phi, g_k^\psi \leftarrow \widehat{\nabla_{\phi,\psi}}\text{EFL}(\phi_k, \psi_k)$ (eq. (10))
  $\phi_{k+1}, \psi_{k+1} \leftarrow \text{UPDATE}(\phi_k, g_k^\phi, \psi_k, g_k^\psi)$
end for
return ϕ_K, ψ_K

Algorithm 1: ACTION-BED

If $p(y_{1:T}|\theta; \pi_d^\phi)$ is suitably reparameterisable, we can optimise this objective using pathwise gradients [55] obtained via reparameterisation [45]:

$$\nabla_{\phi,\psi}\text{EFL}(\phi, \psi) = \mathbb{E}_{p(\theta)q(\varepsilon_{1:T})}\left[\nabla_{\phi,\psi}\ell\big(\pi_a^\psi(h_T(\theta, \varepsilon_{1:T}; \pi_d^\phi)), \theta\big)\right], \tag{9}$$

where we reparameterised $h_T = h_T(\theta, \varepsilon_{1:T}; \pi_d^\phi)$ for $\varepsilon_{1:T} \sim q(\varepsilon_{1:T})$. See Appendix C for full details. We can then estimate these gradients unbiasedly using simple Monte Carlo:

$$\widehat{\nabla_{\phi,\psi}}\,\text{EFL}(\phi, \psi) = \frac{1}{B}\sum_{b=1}^{B}\nabla_{\phi,\psi}\ell\big(\pi_a^\psi(h_T(\theta^b, \varepsilon_{1:T}^b; \pi_d^\phi)), \theta^b\big) \tag{10}$$

for $\theta^b \sim p(\theta), \varepsilon_{1:T}^b \sim q(\varepsilon_{1:T})$, from which stochastic gradient updates can be performed using any desired optimizer. If the simulator is not reparameterisable, we can instead resort to using the REINFORCE estimator [73] to estimate gradients, provided that we have access to the score function of the data generating process. These requirements are the main restrictions of our approach, although we note that a very large class of BED problems satisfy at least one of these assumptions. Details on the architectures used for the policies are provided in Appendix E.

5 Related Work

There are limited prior works considering truly task-driven approaches to BED. One approach is Huang et al. [35], who maximise the improvement in expected utility of the optimal downstream action after observing new data. They only consider utility functions on actions living in outcome space $\mathcal{Y}$, meaning their method cannot be applied as a baseline in any of our BED experiments and is instead applied on prediction-oriented active learning settings [8]. Furthermore, they maximise their expected utility objective by approximating the intractable predictive distribution and explicitly solving the nested optimisation problem over downstream actions. In contrast, our approach does not require any distributional approximations and removes the nested optimisation by learning action policies. Short of being truly task-based, recent works [8, 13, 21, 48, 68, 75, 76] consider "goal-oriented" experimental design, in which designs are sought to maximise the EUR in downstream quantities of interest. While this approach shifts focus from model parameters to quantities of interest, it is still fundamentally an EUR objective and therefore inherits the associated computation challenges. Finally, Kerrigan et al. [44] introduces a new perspective on BED, maximising the dependence between y and θ by solving an optimal transport problem with a chosen cost function. The choice of cost function provides a way of tailoring designs to downstream tasks of interest, but cannot be interpreted as minimising a loss over downstream actions in a principled Bayesian decision-theoretic perspective.

In alternative contexts, Huang et al. [38] introduces a task-driven framework for active learning, which they solve analytically when the downstream loss is a Bregman divergence. This still leads to doubly

intractable objectives, whereas our approach is instead singly intractable for *any* downstream loss. In a Bayesian optimisation (BO) setting, Neiswanger et al. [59] frame classical BO acquisition functions in a loss-based perspective via generalised entropies, which they use to derive new acquisition functions for custom BO tasks. In their case, analytic solutions depend on the Gaussian process posterior, and are thus specific to a particular class of models.

Our work also shares strong connections with reinforcement learning (RL): $\min_{a\in\mathcal{A}} \mathbb{E}_{p(\theta|h_T)}[\ell(a,\theta)]$ can be interpreted as a value function giving the best achievable downstream loss given the current belief state. Prior works [5, 10, 11, 68, 69] have adopted an RL approach to BED by defining rewards as the incremental improvement in the above value function. However, this has only been done for the EIG, implicitly choosing the log-loss to define the value function, while the generalisation to any downstream loss has not yet been made. Furthermore, while variational and amortised inference approaches [12, 20, 26, 37, 68] learn amortised variational approximations which can be interpreted as downstream action networks (often within RL frameworks e.g. Shen et al. [68]), the generalisation to arbitrary downstream actions has also not yet been made. Nonetheless, it is possible our approach could directly benefit from inheriting RL ideas, such as reward shaping and credit assignment, in the future.

6 Experiments

We evaluate **ACTION-BED** across four BED tasks spanning a range of complementary settings. Our experiments illustrate the improved performance and computational efficiency of ACTION-BED compared to baselines. Further evaluations of ACTION-BED in [43] corroborate these findings.

Baselines and evaluation protocol. We compare against three competitive policy-based BED baselines. **DAD** [24] serves as the standard EIG-based policy baseline when an explicit likelihood is available, while **iDAD** [42] provides its likelihood-free counterpart when required. **ALINE** [37] is a state-of-the-art EIG-based approach using a transformer as a joint amortised design and inference network, trained using reinforcement learning with a dense reward. Additionally we include a **Random** policy which samples designs from a task-specific proposal. In order to evaluate these baselines on downstream actions, we train a design policy using the baseline approach, which is then frozen while a downstream action network is trained to minimise the downstream task loss on trajectories generated by the pre-trained design policy. Thus, the design network of each baseline is trained to maximise EIG while the downstream network is trained to minimise downstream loss by approximating the Bayes-optimal downstream action. Methods are evaluated in terms of downstream loss, design policy EIG bounds [24], and end-to-end wall-clock GPU training time.

Training robustness. To confirm that our relative performance gains are not driven by architectural choices, we matched architectures across all baselines and further repeated our experiments using smaller architectures in Appendix G.4. While our main results are reported for a single training seed, we further assess training-seed variability for ACTION-BED and DAD with the main architecture on the source location finding and single pendulum tasks; see Tables 23 and 24. These experiments, together with the multi-seed analysis using smaller architectures in Appendix G.4.2, show that the performance trends are stable across training seeds.

6.1 Source Location Finding

The first experiment is the common source location finding experiment introduced by [24] and inspired by the acoustic energy attenuation model introduced in [70]. The unknown parameter $\theta \in \mathbb{R}^{2\times 2}$ specifies the locations of two sources which emit signals that decay with distance. Over $T = 30$ steps, the design policy sequentially selects locations and observes the resulting aggregate signal intensity. The goal is to learn an adaptive design strategy that enables efficient downstream prediction of source locations. For the downstream predictive loss, we consider the mean squared error (MSE) and the quadratic-logarithmic loss (QLL), a coordinate-wise variant that augments each squared prediction error with a logarithmic penalty before averaging across coordinates. This transformation places greater emphasis on reducing already small errors, thereby encouraging precise localisation.

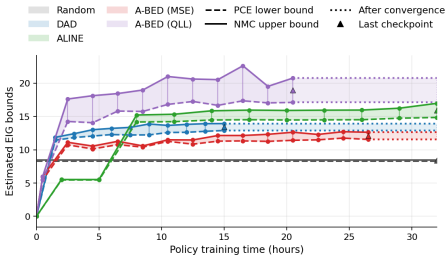


Figure 2: Comparison of EIG bounds as a function of design policy training time, until convergence.

Table 1: **Source Location Finding** final achieved expected losses. Error bars are reported as one standard error over 2048 rollouts. Parentheses indicate the loss used to train the downstream action network. Compute time includes training both the design and downstream action networks.

Method	Performance ($\pm$ 1 s.e.)				Compute
	MSE (10^{-2}) $\downarrow$	QLL $\downarrow$	sPCE $\uparrow$	sNMC $\uparrow$	Time (min) $\downarrow$
Action-BED (MSE)	0.55 ± 0.06	-6.83 ± 0.03	11.63 ± 0.04	12.84 ± 0.08	1171
Action-BED (QLL)	2.8 ± 0.4	$\mathbf{-10.10 \pm 0.03}$	$\mathbf{17.14 \pm 0.11}$	$\mathbf{21.07 \pm 0.45}$	1184
DAD (MSE)	5.6 ± 0.6	-6.35 ± 0.04	12.94 ± 0.03	16.35 ± 0.14	1214
DAD (QLL)	10.7 ± 0.8	-6.58 ± 0.05	12.94 ± 0.03	16.35 ± 0.14	1216
ALINE (MSE)	$\mathbf{0.21 \pm 0.03}$	-7.58 ± 0.05	14.83 ± 0.05	16.97 ± 0.13	2962
ALINE (QLL)	0.81 ± 0.10	-8.11 ± 0.07	14.83 ± 0.05	16.97 ± 0.13	2970
Random	22.4 ± 0.8	-3.16 ± 0.04	8.27 ± 0.04	8.49 ± 0.05	**31**

Both losses are made permutation-invariant over the sources. Implementation and experimental details are provided in Appendix G.1.

Results. Table 1 shows the quantitative results on the source location finding task while qualitative design trajectories are included in Appendix G.1.1. We firstly observe the impressive performance of ACTION-BED under the QLL objective; we achieve both the best downstream QLL error as well as state-of-the-art EIG of the design policy. We believe that this is possible because the QLL remains sensitive to very accurate predictions and continues to reward design policies that identify the sources more precisely, whereas, for example, the sPCE contrastive lower bound saturates at high EIG values therefore becoming harder to train to. A quantile analysis in Table 18 further shows that QLL training yields substantially lower typical localisation errors but a heavier upper tail, explaining its higher mean MSE. Utilising this informative loss function and sidestepping the challenges of EIG-based training is made directly possible by our approach. Under the MSE loss, ALINE performs best however requires around 2.5 times the compute to achieve such a result. A possible explanation for its better performance here is that its dense training rewards help identify the latent sources more efficiently, while such guidance is absent in our setting. ACTION-BED significantly outperforms DAD throughout, which uses a similar amount of compute. In terms of computational efficiency, we show in Figure 2 the attained EIG of the design policy versus training time, where ACTION-BED (QLL) achieves competitive EIG values faster than competing approaches.

Additional ablations and evaluation. We provide several additional analyses in Appendix G.1.2 and G.1.3. We first evaluate the decision-relevant quality of learned design policies independently of their downstream action networks using a common posterior-based decision procedure. This isolates the informativeness of the acquired histories themselves and confirms the objective-dependent ranking observed in the main results. Moreover, it highlights the substantial deployment-time cost avoided by amortised decision-making with ACTION-BED; see Tables 16 and 17. We then investigate the coupling between the jointly learned design policy and downstream action policy in ACTION-BED by evaluating each design and action network cross-pairing; see Table 21. Downstream performance degrades substantially under such policy swaps, showing that the action network exploits not only the observed outcomes but also information encoded in the adaptive design trajectory itself, and can be misled when data is collected by a different policy. This coupling is strongest for the most accurate policies, whose designs concentrate efficiently near the latent sources. We further show for ACTION-BED that retraining the action policy on a fixed pre-trained design policy leads to a mild downstream degradation compared to joint training, suggesting that the joint training dynamics beneficially co-adapt the two policies beyond just ensuring their compatibility; see Table 20. Finally, we ablate the warmstart training of the action network with a random design policy before joint optimisation; see Table 22. Removing it slightly worsens downstream performance, likely because early low-quality action predictions provide a noisier training signal to the design policy, perturbing the joint optimisation dynamics.

6.2 Dynamical Systems

We next consider stochastic single- and double-pendulum systems [6, 41]. In these tasks, the system evolves over time according to a controlled SDE depending on some unknown physical parameters. The continuous dynamics are discretised using Euler-Maruyama over $T = 50$ steps with $dt = 0.05$. At each step an agent influences the dynamics by applying a torque ξ_t. The goal is to learn an adaptive

Table 2: **Stochastic Single Pendulum** final achieved expected losses. Error bars give ± one standard error over 2048 rollouts. Compute time includes training both the design and downstream action networks.

Method	Performance (± 1 s.e.)					Compute
	MSE (10^{-3}) ↓	QLL ↓	W. MSE (10^{-3}) ↓	sPCE ↑	sNMC ↑	Time (min) ↓
Action-BED (MSE)	$\mathbf{6.00 \pm 0.14}$	-6.60 ± 0.05	4.90 ± 0.13	3.68 ± 0.04	3.71 ± 0.04	240
Action-BED (QLL)	7.20 ± 0.17	$\mathbf{-6.75 \pm 0.05}$	4.80 ± 0.11	$\mathbf{3.99 \pm 0.04}$	$\mathbf{4.00 \pm 0.04}$	287
Action-BED (W. MSE)	9.80 ± 0.25	-6.07 ± 0.04	$\mathbf{3.50 \pm 0.09}$	3.33 ± 0.04	3.36 ± 0.04	290
DAD (MSE)	7.20 ± 0.17	-6.20 ± 0.04	5.00 ± 0.14	3.82 ± 0.04	3.88 ± 0.04	670
DAD (QLL)	8.10 ± 0.19	-6.49 ± 0.05	5.20 ± 0.14	3.82 ± 0.04	3.88 ± 0.04	709
DAD (W. MSE)	7.30 ± 0.17	-6.44 ± 0.05	4.40 ± 0.12	3.82 ± 0.04	3.88 ± 0.04	671
ALINE (MSE)	23.50 ± 0.51	-5.43 ± 0.02	7.59 ± 0.12	2.23 ± 0.06	2.24 ± 0.06	2120
ALINE (QLL)	27.53 ± 0.59	-5.69 ± 0.02	9.89 ± 0.17	2.23 ± 0.06	2.24 ± 0.06	2141
ALINE (W. MSE)	23.58 ± 0.50	-5.61 ± 0.02	7.06 ± 0.11	2.23 ± 0.06	2.24 ± 0.06	2122
Random	29.0 ± 1.2	-4.25 ± 0.05	12.6 ± 1.0	1.42 ± 0.03	1.50 ± 0.03	**19**

Table 3: **Stochastic Double Pendulum** final achieved expected losses. Error bars give ± one standard error over 2048 rollouts. Compute time includes training both the design and downstream action networks.

Method	Performance (± 1 s.e.)					Compute
	MSE (10^{-4}) ↓	QLL ↓	W. MSE (10^{-4}) ↓	sPCE ↑	sNMC ↑	Time (min) ↓
Action-BED (MSE)	$\mathbf{1.11 \pm 0.03}$	-10.21 ± 0.04	1.64 ± 0.04	10.73 ± 0.05	12.71 ± 0.19	501
Action-BED (QLL)	1.31 ± 0.03	$\mathbf{-10.32 \pm 0.04}$	1.75 ± 0.05	10.81 ± 0.12	14.29 ± 0.56	504
Action-BED (W. MSE)	1.14 ± 0.03	-10.28 ± 0.04	$\mathbf{1.51 \pm 0.05}$	10.65 ± 0.04	12.10 ± 0.16	504
DAD (MSE)	2.10 ± 0.06	-9.50 ± 0.03	2.47 ± 0.07	$\mathbf{11.37 \pm 0.03}$	$\mathbf{14.79 \pm 0.25}$	934
DAD (QLL)	2.43 ± 0.10	-9.51 ± 0.03	2.79 ± 0.08	$\mathbf{11.37 \pm 0.03}$	$\mathbf{14.79 \pm 0.25}$	940
DAD (W. MSE)	2.13 ± 0.06	-9.47 ± 0.03	2.46 ± 0.07	$\mathbf{11.37 \pm 0.03}$	$\mathbf{14.79 \pm 0.25}$	933
ALINE (MSE)	11.21 ± 0.34	-8.46 ± 0.02	15.12 ± 0.04	9.28 ± 0.07	9.78 ± 0.14	3013
ALINE (QLL)	19.21 ± 0.51	-8.53 ± 0.03	23.29 ± 0.07	9.28 ± 0.07	9.78 ± 0.14	3062
ALINE (W. MSE)	11.81 ± 0.33	-8.52 ± 0.03	14.02 ± 0.04	9.28 ± 0.07	9.78 ± 0.14	3007
Random	20.0 ± 0.9	-6.84 ± 0.05	48.0 ± 1.3	8.26 ± 0.06	8.41 ± 0.08	**58**

design policy that enables accurate downstream prediction of the latent physical parameters $\theta \in \Theta$ from the observed trajectories. We evaluate the prediction with several downstream objectives—MSE, QLL, and a weighted MSE over the components of θ—the latter requiring the design policy to prioritise parameter regions of particular interest. These tasks pose three main challenges: the data-generating process is non-exchangeable due to Markovian dynamics, designs must satisfy physical constraints, and longer rollouts induce complex feedback between interventions and observations.

Stochastic Pendulum. This task comprises a single pendulum with state $y_t = (q_t, \dot{q}_t)$, unknown parameters $\theta \in \Theta \subseteq \mathbb{R}^3$, and bounded control torques $\xi_t \in [-1, 1]$. Full experimental details are provided in Appendix G.2.1. Table 2 confirms that loss-driven acquisition improves the corresponding downstream objective: each variant performs best under the loss used for training. Interestingly, we observe again that the design policy of ACTION-BED (QLL) achieves the highest EIG, outperforming the baselines which directly optimise EIG. We also observe that our method requires substantially less compute than the baselines. It is also notable here that ALINE significantly underperforms even compared to DAD (except on QLL), likely because this problem requires more planning ahead than the earlier location finding experiment.

Stochastic Double-Pendulum. The double-pendulum task is substantially harder as its dynamics are nonlinear, coupled, and potentially chaotic, making trajectories highly sensitive to designs. The state $y_t \in \mathbb{R}^4$ contains the two joint angles and angular velocities, while $\theta \in \Theta \subseteq \mathbb{R}^4$ contains the unknown masses and link lengths. Designs are bounded joint torques $\xi_t = (\xi_t^1, \xi_t^2) \in [-4, 4] \times [-2, 2]$. Details are given in Appendix G.2.2. In this setting, Table 3 again shows that ACTION-BED achieves considerably improved downstream prediction for reduced computational cost. In this task, DAD appears to achieve marginally higher EIG although EIG bounds overlap with ACTION-BED. ALINE again significantly underperfoms compared to the other methods except on QLL.

Table 4: **MNIST Masked Classification**. Error bars give ± one standard error across evaluation batches. We report cross-entropy and classification accuracy on both the training and test datasets. GPU time includes training of both the design and downstream classification networks.

	Train dataset		Test dataset		Compute
Method	Cross-entropy	Accuracy (%)	Cross-entropy	Accuracy (%)	GPU time (min)
Action-BED	$\mathbf{0.033 \pm 0.001}$	$\mathbf{98.87 \pm 0.04}$	$\mathbf{0.097 \pm 0.007}$	$\mathbf{97.42 \pm 0.16}$	9
DAD	0.350 ± 0.004	83.32 ± 0.04	0.546 ± 0.009	81.83 ± 0.16	53
iDAD	0.099 ± 0.002	96.49 ± 0.08	0.164 ± 0.008	95.10 ± 0.21	12
ALINE	0.256 ± 0.008	92.13 ± 0.08	0.271 ± 0.01	91.59 ± 0.10	357
Random	0.786 ± 0.015	71.78 ± 0.87	0.797 ± 0.015	71.40 ± 0.86	**5**

6.3 MNIST Masked Classification

Finally, we consider a sequential masked-classification task adapted from the experimental setting in [39]. Each episode begins with an unobserved image $\theta \in \mathbb{R}^{28\times 28}$ from MNIST [50], together with its class label $z \in \{0, \dots, 9\}$. At each of the $T = 5$ acquisition steps, the policy selects the location $\xi \in [1, 28]^2$ of a 5×5 pixel patch and observes a noisy measurement $y = A_\xi \theta + \eta$, with $\eta \sim \mathcal{N}(0, \sigma^2 I_d)$ and A_ξ the corresponding masking operator. The goal is to select the patches most informative for downstream digit classification. This setting is challenging because acquisition is driven by the class label, whereas measurements arise from the latent image. The patch likelihood $p(y|\theta, \xi)$ is explicit, but its label-conditioned counterpart $p(y|z, \xi)$ is only implicitly induced by the MNIST class distribution. This problem is also strongly information-constrained, since the downstream policy must infer the class from only a few noisy local measurements in a high-dimensional space. Additional details are provided in Appendix G.3.

ACTION-BED naturally targets this implicit problem by jointly training a design policy and downstream classifier to minimise the cross-entropy loss:

$$\min_{\phi,\psi} \mathbb{E}_{p(z)p(\theta|z)p(y_{1:T}|\theta;\pi_d^\phi)} \left[\ell_{\text{CE}}\left(\pi_a^\psi(\cdot|h_T), z\right)\right]$$

where $\ell_{\text{CE}}\left(\pi_a^\psi(\cdot|h_T), z\right) = -\log \pi_a^\psi(z|h_T)$ is the cross-entropy loss and $\pi_a^\psi(\cdot|h_T)$ is a probabilistic classifier. Training to this objective only requires simulating z, θ from the empirical MNIST dataset and reparameterising the simulation of $p(y_{1:T}|\theta; \pi_d)$. By contrast, DAD must target EIG in the latent image θ, since the outcomes conditional on class label $p(y|z, \xi)$ do not have explicit likelihood. IDAD is able to target the EIG in the label z, thereby providing a more competitive baseline. The original ALINE setup does not directly apply in this setting due to the hierarchical likelihood and classification target, we therefore adapted its inference network setup accordingly. Quantitative and qualitative results are reported respectively in Table 4 and Figure 7. ACTION-BED achieves the best downstream performance across the board, demonstrating the lowest cross-entropy loss and highest classification accuracy amongst all approaches, whilst requiring the least training time (other than Random). These results demonstrate that ACTION-BED easily maximises downstream performance in implicit models without additional modifications, critic networks, or internal posterior approximations.

7 Conclusion

In this work, we proposed a paradigm shift in how we think about BED from maximising expected uncertainty reductions (EURs) to minimising the expected future loss (EFL) achieved on downstream tasks of interest. We showed that this EFL formulation follows from first-principles decision theory and recovers the existing uncertainty-based BED construction as an equivalent representation. Critically, though, we showed that the EFL formulation also provides a distinct beneficial mechanism for learning corresponding design policies in practice: by jointly learning a design network and a downstream *action network*, we can recover *singly intractable objectives* for minimising the EFL, whereas EUR formulations are generally doubly intractable. We call this joint training approach ACTION-BED, and show that it provides a practical and effective algorithm for task-driven BED across a variety of problems. ACTION-BED further provides the benefits of naturally allowing implicit models and avoiding ever needing to approximate the full posterior. As such, ACTION-BED further allows us to fully amortise the *entire* decision-making process: not only does it inherit the near-instantaneous design decisions of previous policy-based BED approaches, it further allows the final actions to be chosen with a single forward pass of the learned action network.

Acknowledgements and Disclosures

Funding. Tom Rossa is supported by the EPSRC CDT in Statistics and Machine Learning (EP/Y034813/1), and by GSK Research & Development Limited through a doctoral studentship. Angus Phillips is supported by the EPSRC CDT in Modern Statistics and Statistical Machine Learning (EP/S023151/1). Tom Rainforth is supported by the EPSRC grant EP/Y037200/1 and receives additional research funding from GSK Research & Development Limited.

Competing Interests. The authors declare no additional financial relationships that could reasonably be perceived as influencing this work.

References

[1] Aliprantis, C. D. and Border, K. C. (2006). *Infinite Dimensional Analysis: A Hitchhiker's Guide*. Springer, 3 edition. 17

[2] Ao, Z. and Li, J. (2024). On Estimating the Gradient of the Expected Information Gain in Bayesian Experimental Design. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(18):20311–20319. Number: 18. 2, 3

[3] Ashman, M., Diaconu, C., Weller, A., and Turner, R. E. (2024). In-context in-context learning with transformer neural processes. In Antorán, J. and Naesseth, C. A., editors, *Proceedings of the 6th Symposium on Advances in Approximate Bayesian Inference*, volume 253 of *Proceedings of Machine Learning Research*, pages 1–29. PMLR. 25

[4] Barber, D. and Agakov, F. (2003). Information Maximization in Noisy Channels : A Variational Approach. In *Advances in Neural Information Processing Systems*, volume 16. MIT Press. 2, 3, 5

[5] Barlas, Y. and Salako, K. (2024). Performance comparisons of reinforcement learning algorithms for sequential experimental design. In *Workshop on Generalization in Planning (GenPlan), AAAI 2025*. In Press. 7

[6] Belousov, B., Abdulsamad, H., Schultheis, M., and Peters, J. (2019). Belief space model predictive control for approximately optimal system identification. In *Proceedings of the Multidisciplinary Conference on Reinforcement Learning and Decision Making (RLDM)*. 8

[7] Berger, J. O. (1985). *Statistical Decision Theory and Bayesian Analysis*. Springer Science & Business Media. 3

[8] Bickford Smith, F., Kirsch, A., Farquhar, S., Gal, Y., Foster, A., and Rainforth, T. (2023). Prediction-Oriented Bayesian Active Learning. In *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, pages 7331–7348. PMLR. 6

[9] Bickford Smith, F., Kossen, J., Trollope, E., Van Der Wilk, M., Foster, A., and Rainforth, T. (2025). Rethinking aleatoric and epistemic uncertainty. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 4345–4359. PMLR. 5

[10] Blau, T., Bonilla, E. V., Chades, I., and Dezfouli, A. (2022). Optimizing Sequential Experimental Design with Deep Reinforcement Learning. In *Proceedings of the 39th International Conference on Machine Learning*, pages 2107–2128. PMLR. 3, 7

[11] Blau, T., Chades, I., Dezfouli, A., Steinberg, D. M., and Bonilla, E. V. (2023). Cross-entropy estimators for sequential experiment design with reinforcement learning. In *NeurIPS 2023 Workshop on Adaptive Experimental Design and Active Learning in the Real World*. 7

[12] Bracher, N., Kühmichel, L., Ivanova, D. R., Intes, X., Bürkner, P.-C., and Radev, S. T. (2025). JADAI: Jointly Amortizing Adaptive Design and Bayesian Inference. arXiv:2512.22999 [stat]. 2, 3, 7

[13] Chakraborty, A., Huan, X., and Catanach, T. A. (2024). A Likelihood-Free Approach to Goal-Oriented Bayesian Optimal Experimental Design. 6

[14] Chaloner, K. and Verdinelli, I. (1995). Bayesian Experimental Design: A Review. *Statistical Science*, 10(3):273–304. 1, 2

[15] Coons, T. E. and Huan, X. (2025). A multifidelity estimator of the expected information gain for bayesian optimal experimental design. *SIAM/ASA Journal on Uncertainty Quantification*, 13(4):1990–2021. 3

[16] Cox, R. T. (1946). Probability, frequency and reasonable expectation. *American journal of physics*, 14(1):1–13. 3

[17] Dawid, A. P. (1998). Coherent measures of discrepancy, uncertainty and dependence, with applications to Bayesian predictive experimental design. *Department of Statistical Science, University College London. http://www. ucl. ac. uk/Stats/research/abs94. html, Tech. Rep*, 139. 2, 4, 5

[18] Dawid, A. P. (2007). The geometry of proper scoring rules. *Annals of the Institute of Statistical Mathematics*, 59(1):77–93. 2, 3

[19] DeGroot, M. (1962). Uncertainty, information, and sequential experiments. *The Annals of Mathematical Statistics*, 33(2):404–419. 1, 2, 3, 5

[20] Dong, J., Jacobsen, C., Khalloufi, M., Akram, M., Liu, W., Duraisamy, K., and Huan, X. (2025). Variational Bayesian optimal experimental design with normalizing flows. *Computer Methods in Applied Mechanics and Engineering*, 433:117457. 3, 7

[21] Filstroff, L., Sundin, I., Mikkola, P., Tiulpin, A., Kylmäoja, J., and Kaski, S. (2024). Targeted Active Learning for Bayesian Decision-Making. *Transactions on Machine Learning Research*. 6

[22] Fisher, R. (1935). *The Design of Experiments.* Oliver & Boyd. 2

[23] Fort, G., Gobet, E., and Moulines, E. (2017). Mcmc design-based non-parametric regression for rare event. application to nested risk computations. *Monte Carlo Methods and Applications*, 23(1):21–42. 2

[24] Foster, A., Ivanova, D. R., Malik, I., and Rainforth, T. (2021). Deep Adaptive Design: Amortizing Sequential Bayesian Experimental Design. In *Proceedings of the 38th International Conference on Machine Learning*, pages 3384–3395. PMLR. 2, 3, 6, 7, 25, 31, 32, 50

[25] Foster, A., Jankowiak, M., Bingham, E., Horsfall, P., Teh, Y. W., Rainforth, T., and Goodman, N. (2019). Variational Bayesian Optimal Experimental Design. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc. 2, 3, 5, 20

[26] Foster, A., Jankowiak, M., O'Meara, M., Teh, Y. W., and Rainforth, T. (2020). A unified stochastic gradient approach to designing bayesian-optimal experiments. In Chiappa, S. and Calandra, R., editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 2959–2969. PMLR. 2, 3, 7, 20

[27] Gneiting, T. and Raftery, A. E. (2007). Strictly proper scoring rules, prediction, and estimation. *Journal of the American statistical Association*, 102(477):359–378. 2, 3

[28] Goda, T., Hironaka, T., and Iwamoto, T. (2020). Multilevel Monte Carlo estimation of expected information gains. *Stochastic Analysis and Applications*, 38(4):581–600. 2, 3

[29] Goda, T., Hironaka, T., Kitade, W., and Foster, A. (2022). Unbiased MLMC Stochastic Gradient-Based Optimization of Bayesian Experimental Designs. *SIAM Journal on Scientific Computing*, 44(1):A286–A311. 3

[30] Hedman, M., Ivanova, D. R., Guan, C., and Rainforth, T. (2025). Step-DAD: Semi-Amortized Policy-Based Bayesian Experimental Design. In *Proceedings of the 42nd International Conference on Machine Learning*, pages 22904–22923. PMLR. 3

[31] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780. 25

[32] Huan, X., Jagalur, J., and Marzouk, Y. (2024). Optimal experimental design: Formulations and computations. *Acta Numerica*, 33:715–840. 1, 2

[33] Huan, X. and Marzouk, Y. (2012). Gradient-based stochastic optimization methods in Bayesian experimental design. *International Journal for Uncertainty Quantification*, 4. 3

[34] Huan, X. and Marzouk, Y. M. (2013). Simulation-based optimal Bayesian experimental design for nonlinear systems. *J. Comput. Phys.*, 232(1):288–317. 3

[35] Huang, D., Guo, Y., Acerbi, L., and Kaski, S. (2024). Amortized bayesian experimental design for decision-making. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA. Curran Associates Inc. 2, 3, 6

[36] Huang, D., Huang, Z., Hassan, C., Acerbi, L., Kaski, S., and Rainforth, T. (2026a). Efficient adaptive data acquisition via pretrained belief representations. *Advances in Neural Information Processing Systems*.

[37] Huang, D., Wen, X., Bharti, A., Kaski, S., and Acerbi, L. (2026b). Aline: joint amortization for bayesian inference and active data acquisition. In *Proceedings of the 39th International Conference on Neural Information Processing Systems*, NIPS '25, Red Hook, NY, USA. Curran Associates Inc. 2, 3, 7

[38] Huang, Z., Bickford Smith, F., and Rainforth, T. (2026c). Loss-Driven Bayesian Active Learning. In *International Conference on Artificial Intelligence and Statistics*. 6

[39] Iollo, J., Heinkelé, C., Alliez, P., and Forbes, F. (2025). Bayesian experimental design via contrastive diffusions. In Yue, Y., Garg, A., Peng, N., Sha, F., and Yu, R., editors, *International Conference on Learning Representations*, volume 2025, pages 73125–73151. 2, 3, 10, 46

[40] Iollo, J., Heinkelé, C., Alliez, P., and Forbes, F. (2024). PASOA- PArticle baSed Bayesian Optimal Adaptive design. In *Proceedings of the 41st International Conference on Machine Learning*, pages 21020–21046. PMLR.

[41] Iqbal, S., Corenflos, A., Särkkä, S., and Abdulsamad, H. (2024). Nesting Particle Filters for Experimental Design in Dynamical Systems. In *Proceedings of the 41st International Conference on Machine Learning*, pages 21047–21068. PMLR. 2, 3, 8, 42

[42] Ivanova, D. R., Foster, A., Kleinegesse, S., Gutmann, M. U., and Rainforth, T. (2021). Implicit Deep Adaptive Design: Policy-Based Experimental Design without Likelihoods. In *Advances in Neural Information Processing Systems*, volume 34, pages 25785–25798. Curran Associates, Inc. 3, 6, 7, 25, 52

[43] Kerrigan, G., Hedman, M., Iqbal, S., and Rainforth, T. (2026). TestBED: An open-source toolkit for bayesian experimental design. arXiv preprint arXiv:2611.06633. 2, 7

[44] Kerrigan, G., Naesseth, C. A., and Rainforth, T. (2025). A Geometric Approach to Optimal Experimental Design. arXiv:2510.14848 [stat]. 1, 6

[45] Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. In *International Conference on Learning Representations*. 6

[46] Kleinegesse, S. and Gutmann, M. U. (2019). Efficient Bayesian Experimental Design for Implicit Models. In *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, pages 476–485. PMLR. 3

[47] Kleinegesse, S. and Gutmann, M. U. (2020). Bayesian Experimental Design for Implicit Models by Mutual Information Neural Estimation. In *Proceedings of the 37th International Conference on Machine Learning*, pages 5316–5326. PMLR. 3

[48] Kleinegesse, S. and Gutmann, M. U. (2021). Gradient-based Bayesian Experimental Design for Implicit Models using Mutual Information Lower Bounds. arXiv:2105.04379 [stat]. 6

[49] Kuratowski, K. and Ryll-Nardzewski, C. (1965). A general theorem on selectors. *Bull. Acad. Polon. Sci. Sér. Sci. Math. Astronom. Phys*, 13(6):397–403. 17

[50] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324. 10, 46

[51] Lee, J., Lee, Y., Kim, J., Kosiorek, A. R., Choi, S., and Teh, Y. W. (2019). Set transformer: A framework for attention-based permutation-invariant neural networks. In *Proceedings of the 36th International Conference on Machine Learning*. 25

[52] Lindley, D. V. (1956). On a Measure of the Information Provided by an Experiment. *The Annals of Mathematical Statistics*, 27(4):986–1005. 1, 2

[53] Lindley, D. V. (1972). *Bayesian statistics*. Society for Industrial and Applied Mathematics. 1, 2, 3, 4

[54] Lindley, D. V. (1982). Scoring rules and the inevitability of probability. *International statistical review/revue internationale de statistique*, pages 1–11. 2, 3

[55] Mohamed, S., Rosca, M., Figurnov, M., and Mnih, A. (2020). Monte Carlo Gradient Estimation in Machine Learning. *Journal of Machine Learning Research*, 21(132):1–62. 6

[56] Murphy, K. (2024). Reinforcement learning: an overview. *arXiv preprint arXiv:2412.05265*. 4

[57] Myung, J. I., Cavagnaro, D. R., and Pitt, M. A. (2013). A tutorial on adaptive design optimization. *Journal of Mathematical Psychology*, 57(3):53–67. 3

[58] Neal, R. M. (2011). Mcmc using hamiltonian dynamics. In Brooks, S., Gelman, A., Jones, G. L., and Meng, X.-L., editors, *Handbook of Markov Chain Monte Carlo*, pages 113–162. Chapman and Hall/CRC. 36

[59] Neiswanger, W., Yu, L., Zhao, S., Meng, C., and Ermon, S. (2022). Generalizing Bayesian optimization with decision-theoretic entropies. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, pages 21016–21029, Red Hook, NY, USA. Curran Associates Inc. 7

[60] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32 (NeurIPS)*, pages 8024–8035. 31

[61] Phillips, A., Kerrigan, G., and Rainforth, T. (2026). Bayesian experimental design via score matching. In Perković, E. and Malinsky, D., editors, *Proceedings of the 42nd Conference on Uncertainty in Artificial Intelligence*, volume 337 of *Proceedings of Machine Learning Research*, pages 5393–5426. PMLR. 2, 3, 4

[62] Raiffa, H. and Schlaifer, R. (1961). *Applied statistical decision theory*. John Wiley & Sons. 3

[63] Rainforth, T., Cornish, R., Yang, H., Warrington, A., and Wood, F. (2018). On Nesting Monte Carlo Estimators. In *Proceedings of the 35th International Conference on Machine Learning*, pages 4267–4276. PMLR. 2, 3

[64] Rainforth, T., Foster, A., Ivanova, D. R., and Bickford Smith, F. (2024). Modern Bayesian Experimental Design. *Statistical Science*, 39(1):100–114. 1, 2

[65] Savage, L. J. (1951). *The theory of statistical decision*, volume 46. Journal of the American Statistical Association. 1, 3

[66] Savage, L. J. (1971). Elicitation of personal probabilities and expectations. *Journal of the American Statistical Association*, 66(336):783–801. 2, 3

[67] Shannon, C. E. (1948). A Mathematical Theory of Communication. *Bell System Technical Journal*, 27(3):379–423. 1

[68] Shen, W., Dong, J., and Huan, X. (2025). Variational sequential optimal experimental design using reinforcement learning. *Computer Methods in Applied Mechanics and Engineering*, 444:118068. 2, 3, 6, 7

[69] Shen, W. and Huan, X. (2023). Bayesian Sequential Optimal Experimental Design for Nonlinear Models Using Policy Gradient Reinforcement Learning. *Computer Methods in Applied Mechanics and Engineering*, 416:116304. arXiv:2110.15335 [cs]. 3, 7

[70] Sheng, X. and Hu, Y.-H. (2005). Maximum likelihood multiple-source localization using acoustic energy measurements with wireless sensor networks. *IEEE Transactions on Signal Processing*, 53(1):44–53. 7

[71] von Neumann, J. and Morgenstern, O. (1944). *Theory of Games and Economic Behavior*. Princeton University Press, Princeton, NJ. 3

[72] Wald, A. (1950). Statistical decision functions. In *Breakthroughs in Statistics: Foundations and Basic Theory*, pages 342–357. Springer. 3

[73] Williams, R. J. (1992). Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Mach. Learn.*, 8(3-4):229–256. 6

[74] Zaheer, M., Kottur, S., Ravanbakhsh, S., Póczos, B., Salakhutdinov, R., and Smola, A. J. (2017). Deep sets. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30 (NeurIPS 2017)*, pages 3391–3401. Curran Associates, Inc. 25

[75] Zhang, Z., Dong, J., Liu, J., and Huan, X. (2025). Goal-oriented sequential bayesian experimental design for causal learning. 6

[76] Zhong, S., Shen, W., Catanach, T., and Huan, X. (2026). Goal-Oriented Bayesian Optimal Experimental Design for Nonlinear Models using Markov Chain Monte Carlo. *SIAM/ASA Journal on Uncertainty Quantification*, 14(1):19–47. arXiv:2403.18072 [stat]. 6

A Lifting Bayes-Optimal Actions to Action Policies

In this section, we formalise the equivalence between the nested optimisation over Bayes-optimal downstream actions in Definition 1 and the joint optimisation over design and action policies used in Equation (5). The key point is that, under standard measurability and integrability assumptions, a pointwise optimisation over actions after observing the experimental history can be lifted to an optimisation over measurable action policies. We establish this equivalence in the more general setting where the downstream loss may depend explicitly on the observed history h_T, a dependence omitted in the main text for simplicity.

Setup. Let $(\Omega, \mathcal{F}, \mathbb{P})$ be the underlying probability space, and $\pi_d \in \Pi_d$ be a fixed design policy. We assume that h_T takes values in a measurable history space $(\mathcal{H}_T, \mathcal{B}(\mathcal{H}_T))$. We consider a downstream loss $\ell : \mathcal{A} \times \Theta \times \mathcal{H}_T \to \bar{\mathbb{R}}$, which may depend explicitly on the observed history. For a fixed history $h \in \mathcal{H}_T$, we formally define the conditional risk $\forall a \in \mathcal{A}$ as,

$$L(a, h) := \mathbb{E}[\ell(a, \theta, h) \mid h_T = h] .$$

The Bayes-optimal action after observing h is any solution of

$$a^B(h) \in \arg\min_{a \in \mathcal{A}} L(a, h).$$

The corresponding Bayes expected future loss for a design policy π_d is

$$\mathrm{EFL}_\ell^B(\pi_d) = \mathbb{E}_{p(h_T;\pi_d)} \left[\min_{a \in \mathcal{A}} L(a, h_T) \right].$$

We now show that this nested formulation is equivalent to an optimisation over measurable action policies.

Standing assumptions. We work under standard regularity assumptions ensuring that the pointwise Bayes-optimal actions can be selected measurably as functions of the observed history. The history space $(\mathcal{H}_T, \mathcal{B}(\mathcal{H}_T))$ is standard Borel, and the action space $\mathcal{A}$ is a compact metric space with Borel σ-algebra $\mathcal{B}(\mathcal{A})$. For every $\pi_d \in \Pi_d$, the conditional risk $L(a, h)$ admits a jointly measurable version on $\mathcal{A} \times \mathcal{H}_T$, and $a \mapsto L(a, h)$ is lower semicontinuous for every $h \in \mathcal{H}_T$. We also assume the relevant expectations are well-defined, for example through an integrable envelope $m(h_T, \theta)$ satisfying

$$\forall (a, \theta, h_T), \quad |\ell(a, \theta, h_T)| \le m(h_T, \theta) \quad \wedge \quad \mathbb{E}[m(h_T, \theta)] < \infty.$$

Let

$$\Pi_a^{\mathrm{meas}} := \{\pi_a : \mathcal{H}_T \to \mathcal{A} \ : \ \pi_a \text{ is } \mathcal{B}(\mathcal{H}_T)/\mathcal{B}(\mathcal{A})\text{-measurable}\} \tag{11}$$

denote the set of admissible measurable action policies. The restriction to measurable action policies makes the optimisation probabilistically well-defined. Under the regularity assumptions above, the Bayes-action correspondence admits a measurable selector, so the pointwise Bayes action belongs to Π_a^{meas}. It yields the desired equivalence, as stated in the following proposition.

Proposition 1 (Lifting pointwise Bayes actions to action policies)**.** *Under the previous assumptions, for every fixed design policy $\pi_d \in \Pi_d$,*

$$\mathbb{E}_{p(h_T;\pi_d)} \left[\min_{a \in \mathcal{A}} L(a, h_T) \right] = \min_{\pi_a \in \Pi_a^{\mathrm{meas}}} \mathbb{E}_{p(h_T;\pi_d)} \left[L(\pi_a(h_T), h_T) \right]. \tag{12}$$

Equivalently,

$$\mathrm{EFL}_\ell^B(\pi_d) = \min_{\pi_a \in \Pi_a^{\mathrm{meas}}} \mathbb{E}_{p(h_T;\pi_d)p(\theta|h_T)} \left[\ell(\pi_a(h_T), \theta, h_T) \right]. \tag{13}$$

Proof. Fix $\pi_d \in \Pi_d$. For any measurable action policy $\pi_a \in \Pi_a^{\mathrm{meas}}$, we directly have pointwise

$$\min_{a \in \mathcal{A}} L(a, h) \le L(\pi_a(h), h).$$

Therefore,

$$\mathbb{E}_{p(h_T;\pi_d)} \left[\min_{a \in \mathcal{A}} L(a, h_T) \right] \le \mathbb{E}_{p(h_T;\pi_d)} \left[L(\pi_a(h_T), h_T) \right].$$

Taking the infimum over $\pi_a \in \Pi_a^{\text{meas}}$ gives

$$\mathbb{E}_{p(h_T;\pi_d)}\left[\min_{a\in\mathcal{A}} L(a,h_T)\right] \leq \inf_{\pi_a\in\Pi_a^{\text{meas}}} \mathbb{E}_{p(h_T;\pi_d)}\left[L(\pi_a(h_T),h_T)\right].$$

It remains to prove the reverse inequality. Since $L(\cdot,h)$ is lower semicontinuous on the compact action space $\mathcal{A}$, the lower semicontinuous version of Weierstrass' theorem implies that $\arg\min_{a\in\mathcal{A}} L(a,h)$ is non-empty for every $h \in \mathcal{H}_T$. Hence the argmin correspondence

$$\Gamma(h) := \arg\min_{a\in\mathcal{A}} L(a,h)$$

is non-empty and compact-valued, as a closed subset of $\mathcal{A}$. Since L is jointly measurable and lower semicontinuous in the action variable, the measurable maximum theorem from [1] implies that the value function

$$h \mapsto \min_{a\in\mathcal{A}} L(a,h)$$

is measurable and that Γ is a measurable correspondence. By the Kuratowski–Ryll-Nardzewski measurable selection theorem from [49], there exists a measurable selector $\pi_a^B : \mathcal{H}_T \to \mathcal{A}$ such that

$$\pi_a^B(h) \in \Gamma(h) \qquad \text{for all } h \in \mathcal{H}_T.$$

Therefore, $\forall h \in \mathcal{H}_T$

$$L(\pi_a^B(h),h) = \min_{a\in\mathcal{A}} L(a,h),$$

and hence

$$\mathbb{E}_{p(h_T;\pi_d)}\left[L(\pi_a^B(h_T),h_T)\right] = \mathbb{E}_{p(h_T;\pi_d)}\left[\min_{a\in\mathcal{A}} L(a,h_T)\right].$$

Thus the infimum over measurable policies is attained by π_a^B, giving the desired equality. □

From Bayes-action policies to joint design–action optimisation. Having shown the equality (13), we can directly rewrite it as

$$\text{EFL}_\ell^B(\pi_d) = \min_{\pi_a\in\Pi_a^{\text{meas}}} \mathbb{E}_{p(\theta)p(y_{1:T}|\theta;\pi_d)}\left[\ell(\pi_a(h_T),\theta,h_T)\right]. \tag{14}$$

Consequently, optimising the Bayes expected future loss over design policies can be written as

$$\begin{aligned}\pi_d^B &\in \arg\min_{\pi_d\in\Pi_d} \text{EFL}_\ell^B(\pi_d) \\ &= \arg\min_{\pi_d\in\Pi_d} \min_{\pi_a\in\Pi_a^{\text{meas}}} \mathbb{E}_{p(\theta)p(y_{1:T}|\theta;\pi_d)}\left[\ell(\pi_a(h_T),\theta,h_T)\right],\end{aligned}$$

which ultimately leads to the core objective used by ACTION-BED as described in equation (5).

Remark. The compactness and lower semicontinuity assumptions above are standard sufficient conditions ensuring that exact Bayes actions exist and admit measurable selectors, which is enough for the equivalence used in this work. When these assumptions fail, the result can be stated in a weaker but more general form using infima and measurable ε-optimal selectors.

Specifically, if for every $\varepsilon > 0$ there exists a measurable policy $\pi_a^\varepsilon : \mathcal{H}_T \to \mathcal{A}$ satisfying

$$L(\pi_a^\varepsilon(h),h) \leq \inf_{a\in\mathcal{A}} L(a,h) + \varepsilon,$$

then

$$\mathbb{E}_{p(h_T;\pi_d)}\left[\inf_{a\in\mathcal{A}} L(a,h_T)\right] = \inf_{\pi_a\in\Pi_a^{\text{meas}}} \mathbb{E}_{p(h_T;\pi_d)}\left[L(\pi_a(h_T),h_T)\right].$$

This extension is useful when the action space is non-compact or when the conditional risk is not known to attain its pointwise infimum.

B Equivalence Between BED Viewpoints

In the following Section we expand on the discussion on the equivalence between EUR and EFL perspectives of BED by providing a proof of Theorem 1. In section B.2 we additionally elaborate on the equivalence between ACTION-BED and existing variational approaches in a particular case.

B.1 Proof of Theorem 1

Assumptions. Let Θ and $\mathcal{H}_T$ be standard Borel spaces, and let $\mathcal{P}_\theta$ be a measurable class of probability distributions on Θ containing the posterior distributions $p(\theta \mid y_{1:T}; \pi_d)$, up to null sets, for every admissible design policy. We assume that all scoring rules considered are jointly measurable and proper for every fixed history $h \in \mathcal{H}_T$. We assume that the posterior expected self-scores, Bayes risks, and their expectations over experimental histories are finite for every admissible design policy. For any $p, q \in \mathcal{P}_\theta$ and history h, cross-score expectations $\mathbb{E}_{\theta\sim p}[s(q, h, \theta)]$ are assumed to be well-defined, possibly taking the value $+\infty$. In particular, all comparisons below are understood in the extended real numbers without indeterminate expressions.

For a downstream loss ℓ, we assume that $\ell(a, \theta, h)$ is jointly measurable and that the corresponding Bayes risks are well-defined. Whenever Bayes-optimal actions are invoked, we assume that the relevant infima are attained. For the construction of a scoring rule from a downstream loss, we additionally assume that, for every $(q_\theta, h) \in \mathcal{P}_\theta \times \mathcal{H}_T$, the Bayes-action set is non-empty and admits a jointly measurable selector

$$\tilde{\pi}_a^B(h, q_\theta) \in \arg\min_{a\in\mathcal{A}} \mathbb{E}_{\theta\sim q_\theta}[\ell(a, \theta, h)].$$

Unlike Appendix A, we do not require compactness of the action space or lower semicontinuity of the loss. The conditions of Appendix A provide sufficient conditions for measurable Bayes-action selection in the policy-lifting result, but are not required for the equivalence established here.

We restate the theorem for completeness in a more general form, allowing both the downstream loss and the scoring rule to depend explicitly on the observed data.

Theorem 1 (Equivalence of EFL & EPU)**.** *Let $s : \mathcal{P}_\theta \times \mathcal{H}_T \times \Theta \to \bar{\mathbb{R}}$ be a proper scoring rule on Θ for every fixed history $h_T \in \mathcal{H}_T$. Then any*

$$\pi_d^B \in \arg\min_{\pi_d\in\Pi_d} \mathbb{E}_{p(\theta)p(y_{1:T}|\theta;\pi_d)}\big[s\big(p_\theta(\cdot \mid h_T), h_T, \theta\big)\big] \tag{15}$$

is a Bayes-optimal design policy for the belief model $p(\theta)p(y_{1:T}|\theta; \pi_d)$ and for any loss $\ell \in \mathcal{L}_s$, where

$$\mathcal{L}_s = \Big\{\ell : \forall (q_\theta, h) \in \mathcal{P}_\theta \times \mathcal{H}_T,\ \inf_{a\in\mathcal{A}} \mathbb{E}_{\theta\sim q_\theta}\big[\ell(a, \theta, h)\big] = \mathbb{E}_{\theta\sim q_\theta}\big[s(q_\theta, h, \theta)\big]\Big\}, \tag{16}$$

Furthermore, the set $\mathcal{L}_s$ is never empty, and any loss function ℓ satisfying the previous integrability conditions and admitting measurable Bayes-action selectors induces one or more corresponding proper scoring rules s, though not necessarily strictly proper, of the form

$$s(q_\theta, h, \theta) = \ell\big(\tilde{\pi}_a^B(h, q_\theta), \theta, h\big), \tag{17}$$

where

$$\tilde{\pi}_a^B(h, q_\theta) \in \arg\min_{a\in\mathcal{A}} \mathbb{E}_{\theta\sim q_\theta}[\ell(a, \theta, h)]. \tag{18}$$

Here, the action space $\mathcal{A}$ is understood to be part of the specification of each loss function ℓ. If the Bayes-action set contains multiple elements for some (h, q_θ), different measurable selections may induce distinct proper scoring rules for the same loss function. All expectations and minimizers are assumed to be well-defined.

Proof. We start by noting that, for any $\ell \in \mathcal{L}_s$,

$$\mathbb{E}_{p(\theta)p(y_{1:T}|\theta;\pi_d)}\left[s(p_\theta(\cdot \mid h_T), h_T, \theta)\right] = \mathbb{E}_{p(h_T;\pi_d)}\left[\mathbb{E}_{p(\theta|h_T)}\left[s(p_\theta(\cdot \mid h_T), h_T, \theta)\right]\right] \tag{19}$$

$$= \mathbb{E}_{p(h_T;\pi_d)}\left[\inf_{a\in\mathcal{A}} \mathbb{E}_{p(\theta|h_T)}\left[\ell(a, \theta, h_T)\right]\right]. \tag{20}$$

The last equality follows directly from the definition of $\mathcal{L}_s$. Therefore, for any $\ell \in \mathcal{L}_s$, the minimizers of Equation (15) coincide with the Bayes-optimal experimental design policies associated with ℓ,

as defined in Definition 1, allowing for explicit dependence on h_T. Hence, π_d^B is a Bayes-optimal experimental design policy for ℓ under the belief model $p(\theta)p(y_{1:T} \mid \theta; \pi_d)$.

Next, we show that for any ℓ satisfying the stated assumptions, any corresponding s defined as per Equation (17) must be a proper scoring rule for all $h_T \in \mathcal{H}_T$. To do this, let $\tilde{\pi}_a^B : \mathcal{H}_T \times \mathcal{P}_\theta \to \mathcal{A}$ denote an action policy that deterministically returns, for any given data and belief state (h_T, p_θ), a corresponding Bayes-optimal action as per Equation (18). We note that different choices for $\tilde{\pi}_a^B$ may lead to different s. By the definition of s, we have

$$\inf_{q_\theta \in \mathcal{P}_\theta} \mathbb{E}_{p(\theta|y_{1:T};\pi_d)} \left[s(q_\theta, h_T, \theta) \right] = \inf_{q_\theta \in \mathcal{P}_\theta} \mathbb{E}_{p(\theta|y_{1:T};\pi_d)} \left[\ell\big(\tilde{\pi}_a^B(h_T, q_\theta), \theta, h_T\big) \right].$$

Since q_θ influences the expected loss only through the choice of action $\tilde{\pi}_a^B(h_T, q_\theta)$, it follows that

$$\inf_{q_\theta \in \mathcal{P}_\theta} \mathbb{E}_{p(\theta|y_{1:T};\pi_d)} \left[\ell\big(\tilde{\pi}_a^B(h_T, q_\theta), \theta, h_T\big) \right] \geq \inf_{a \in \mathcal{A}} \mathbb{E}_{p(\theta|y_{1:T};\pi_d)} \left[\ell(a, \theta, h_T) \right].$$

Finally, using the definition of $\tilde{\pi}_a^B$ that returns correspondingly Bayes-optimal actions, the right-hand side is attained by choosing the true posterior belief $q_\theta = p_\theta(\cdot \mid y_{1:T}; \pi_d)$, so that

$$\begin{aligned} \inf_{a \in \mathcal{A}} \mathbb{E}_{p(\theta|y_{1:T};\pi_d)} \left[\ell(a, \theta, h_T) \right] &= \mathbb{E}_{p(\theta|y_{1:T};\pi_d)} \left[\ell(\tilde{\pi}_a \left(h_T, p_\theta(\cdot \mid y_{1:T}; \pi_d) \right), \theta, h_T) \right] \\ &= \mathbb{E}_{p(\theta|y_{1:T};\pi_d)} \left[s(p_\theta(\cdot \mid y_{1:T}; \pi_d), h_T, \theta) \right]. \end{aligned}$$

Thus, $p_\theta(\cdot \mid y_{1:T}; \pi_d)$ is a minimizer of $\mathbb{E}_{p(\theta|y_{1:T};\pi_d)} \left[s(q_\theta, h_T, \theta) \right]$, and so s satisfies the definition of a proper scoring rule on Θ. It need not be strictly proper as the minimiser is not necessarily unique.

Finally, we show that $\mathcal{L}_s$ can never be an empty set for a given scoring rule s. Starting from Equation (15), we have

$$\pi_d^B = \arg\min_{\pi_d \in \Pi_d} \mathbb{E}_{p(\theta)p(y_{1:T}|\theta;\pi_d)} \left[s(p_\theta(\cdot \mid y_{1:T}; \pi_d), h_T, \theta) \right] \tag{21}$$

$$= \arg\min_{\pi_d \in \Pi_d} \mathbb{E}_{p(y_{1:T};\pi_d)} \left[\mathbb{E}_{p(\theta|y_{1:T};\pi_d)} \left[s(p_\theta(\cdot \mid y_{1:T}; \pi_d), h_T, \theta) \right] \right]. \tag{22}$$

Since s is a proper scoring rule, this is equal to

$$\pi_d^B = \arg\min_{\pi_d \in \Pi_d} \mathbb{E}_{p(y_{1:T};\pi_d)} \left[\min_{q_\theta \in \mathcal{P}_\theta} \mathbb{E}_{p(\theta|y_{1:T};\pi_d)} \left[s(q_\theta, h_T, \theta) \right] \right]. \tag{23}$$

This is the general form of a Bayes-optimal experimental design policy with action space $\mathcal{A} = \mathcal{P}_\theta$ and downstream loss

$$\ell(q_\theta, \theta, h_T) = s(q_\theta, h_T, \theta).$$

Therefore, s is itself an element of $\mathcal{L}_s$, and consequently $\mathcal{L}_s$ can never be empty. □

B.2 Recovering Variational Approaches

As noted in Section 3.3, the EIG itself is equivalent, up to an additive constant and a change of sign, to an EPU objective with the log score $s(q_\theta(\cdot), \theta) = -\log q_\theta(\theta)$. That is, up to a constant offset, the EIG is equivalent to

$$\mathbb{E}_{p(\theta)p(y_{1:T}|\theta;\pi_d)} \left[-\log p(\theta|h_T) \right].$$

We can find an equivalent EFL formulation by choosing $\ell \in \mathcal{L}_s$ according to Theorem 1. The canonical choice of loss function $\ell \in \mathcal{L}_s$ for the log score is the log-loss on actions $a(\cdot) \in \mathcal{P}_\theta$, that is $\ell(a(\cdot), \theta) = -\log a(\theta)$. The optimal action under this loss is a standard result:

$$\tilde{\pi}_a = \arg\min_{a(\cdot) \in \mathcal{P}_\theta} \mathbb{E}_{\theta \sim p(\cdot|h_T)} \left[-\log a(\theta) \right] = \arg\min_{a(\cdot) \in \mathcal{P}_\theta} \mathrm{KL}(p(\theta|h_T) \| a(\theta)) = p(\theta|h_T),$$

i.e. true Bayesian posterior. Therefore ACTION-BED solves the following joint optimisation problem:

$$\begin{aligned} (\pi_a, \pi_d)^B = &\arg\min_{(\pi_a, \pi_d) \in \Pi_a^{\mathcal{P}_\theta} \times \Pi_d} \mathrm{EFL}_{\text{log-loss}}(\pi_d, \pi_a), \\ \text{where} \quad \mathrm{EFL}_{\text{log-loss}}(\pi_d, \pi_a) &= \mathbb{E}_{p(\theta)p(y_{1:T}|\theta;\pi_d)} \left[-\log(\pi_a(h_T)(\theta)) \right], \end{aligned} \tag{24}$$

and $\Pi_a^{\mathcal{P}_\theta} = \{\pi \mid \pi : \mathcal{Y}^T \times \Xi^T \to \mathcal{P}_\theta\}$. In practice, we must parameterise $\Pi_a^{\mathcal{P}_\theta}$, or a subset thereof, in order to pose a tractable parametric optimisation problem. We do this by specifying an amortised posterior approximation *policy* $\pi_a^\psi : \mathcal{Y}^T \times \Xi^T \to \mathcal{P}_\theta$, defined as:

$$\pi_a^\psi(h_T)(\cdot) = q_\psi(\cdot|h_T),$$

where $\psi \in \Psi$ parametrises an amortised variational family. Thus, ACTION-BED solves the following parametric optimisation:

$$\min_{\phi,\psi} \mathbb{E}_{p(\theta)p(y_{1:T}|\theta;\pi_d^\phi)} \left[-\log \pi_a^\psi(h_T)(\theta)\right] = \min_{\phi,\psi} \mathbb{E}_{p(\theta)p(y_{1:T}|\theta;\pi_d^\phi)} \left[-\log q_\psi(\theta|h_T)\right].$$

This objective is equivalent to the variational posterior objectives introduced by Foster et al. [25, 26], who solve the following problem:

$$\max_{\phi,\psi} \mathbb{E}_{p(\theta)p(y_{1:T}|\theta;\pi_d^\phi)} \left[\log q_\psi(\theta|h_T) - \log p(\theta)\right].$$

C Reparameterisation of Simulator and Regularity Conditions

In this Section, we provide details of the reparameterisation function used in our EFL gradient expression Equation (9). We further provide the precise regularity conditions required to differentiate under the integral sign in the reparameterisation trick.

We recall the application of the reparameterisation trick as follows:

$$\nabla_{\phi,\psi}\text{EFL}(\phi,\psi) = \nabla_{\phi,\psi}\mathbb{E}_{p(\theta)\,p(y_{1:T}|\theta;\pi_d^\phi)}\left[\ell\big(\pi_a^\psi(h_T),\theta\big)\right] \tag{25}$$

$$= \mathbb{E}_{p(\theta)q(\varepsilon_{1:T})}\left[\nabla_{\phi,\psi}\ell\big(\pi_a^\psi(h_T(\theta,\varepsilon_{1:T};\pi_d^\phi)),\theta\big)\right], \tag{26}$$

where the simulator $p(y_{1:T} \mid \theta;\pi_d^\phi)$ was reparameterised in order to write $h_T = h_T(\theta,\varepsilon_{1:T};\pi_d^\phi)$ for $\varepsilon_{1:T} \sim q(\varepsilon_{1:T})$. We firstly provide the exact form of the reparameterisation function in section C.1 before providing regularity conditions on the loss and reparameterisation functions in section C.2 and consequently on the BED model in section C.3.

C.1 Reparametrisation Functions

Recall the data generating process under a given design policy π_d^ϕ is given by $p(y_{1:T} \mid \theta;\pi_d^\phi) = \prod_{t=1}^T p(y_t \mid \theta, \xi_t = \pi_d^\phi(h_{t-1}), h_{t-1})$. Furthermore recall the data h_T is defined as $h_T = \{y_{1:T}, \xi_{1:T}\}$ and designs ξ_t are recovered auto-regressively using the design policy and the previous data $\xi_t = \pi_d^\phi(h_{t-1})$.

We can therefore build the reparameterisation of h_T recursively as follows. At the first step, we have $h_1 = \{y_1, \xi_1\}$ where

$$y_1 \sim p(\cdot \mid \theta, \xi_1), \quad \xi_1 = \pi_d^\phi(\emptyset).$$

Introducing noise variable $\varepsilon_1 \sim q_1(\cdot)$, we can directly write the pair (y_1, ξ_1) as a function of $(\theta, \epsilon_1;\pi_d^\phi)$, specifically:

$$(y_1,\xi_1) = \Big(\tilde{g}(\theta,\epsilon_1;\pi_d^\phi), \pi_d^\phi(\emptyset)\Big) =: h_1(\theta,\epsilon_1;\pi_\phi^d).$$

At the second time step, we now have

$$y_2 \sim p(\cdot \mid \theta, \xi_2, h_1), \quad \xi_2 = \pi_d^\phi(h_1).$$

Again introducing noise variable $\varepsilon_2 \sim q_2(\cdot)$, we can express the pair (y_2, ξ_2) as a function of $(\theta, \epsilon_2, h_1;\pi_d^\phi)$ as follows:

$$(y_2,\xi_2) = \Big(\tilde{g}(\theta,\epsilon_2,h_1;\pi_d^\phi), \pi_d^\phi(h_1)\Big) =: \tilde{h}_2(\theta,\varepsilon_2,h_1;\pi_d^\phi)$$

By substituting the reparameterisation of the previous step, i.e. $h_1 = h_1(\theta,\epsilon_1;\pi_\phi^d)$ for $\varepsilon_1 \sim q_1(\cdot)$ and stacking with h_1, we obtain:

$$h_2 = (y_1,\xi_1,y_2,\xi_2) = (h_1, y_2, \xi_2) = (h_1, \tilde{h}_2) =: h_2(\theta,\epsilon_{1:2};\pi_d^\phi).$$

We continue this argument recursively, arriving at a reparameterisation function for the entire experimental history $h_T = h_T(\theta,\epsilon_{1:T};\pi_d^\phi)$ for $\varepsilon_{1:T} \sim \prod_{t=1}^T q_t(\cdot)$. Note that we may sometimes abuse notation slightly and write this as $h_T = h_T(\theta,\varepsilon_{1:T},\phi)$, with the dependence on the policy π_d implicit.

C.2 Conditions for Differentiation Under the Integral Sign

Let $(\Omega, \mathcal{F}, \mu)$ be a measure space on which the random variables θ and ε are defined. Let $\Phi \times \Psi \subset \mathbb{R}^{d_\phi} \times \mathbb{R}^{d_\psi}$ be an open set. Equation 26 is justified under the following conditions:

1. The distributions $p(\theta)$ and $p(\varepsilon_{1:T})$ are independent of (ϕ,ψ);
2. For μ-almost every $(\theta,\varepsilon_{1:T}) \in \Omega$, the mapping

$$(\phi,\psi) \mapsto \ell\big(\pi_a^\psi(h_T(\theta,\varepsilon_{1:T};\pi_d^\phi)),\theta\big)$$

is $\mathcal{F}$-measurable and integrable:

$$\int_\Omega \left|\ell\big(\pi_a^\psi(h_T(\theta,\varepsilon_{1:T};\pi_d^\phi)),\theta\big)\right| \,\mathrm{d}\mu(\theta,\varepsilon_{1:T}) < \infty.$$

3. For μ-almost every $(\theta, \varepsilon_{1:T}) \in \Omega$, the mapping is differentiable in (ϕ, ψ) and the partial derivatives
$$\partial_\phi \ell\big(\pi_a^\psi(h_T(\theta, \varepsilon_{1:T}; \pi_d^\phi)), \theta\big), \quad \partial_\psi \ell\big(\pi_a^\psi(h_T(\theta, \varepsilon_{1:T}; \pi_d^\phi)), \theta\big)$$
exist (which holds under standard smoothness assumptions on h_T, π_a^ψ, and ℓ);
4. For each compact $K \subset \Phi \times \Psi$, there exist integrable functions
$$g_\phi : \Omega \to [0, \infty), \quad g_\psi : \Omega \to [0, \infty)$$
such that for all $(\phi, \psi) \in K$ and μ-almost every $(\theta, \varepsilon_{1:T}) \in \Omega$,
$$\big|\partial_\phi\, \ell\big(\pi_a^\psi(h_T(\theta, \varepsilon_{1:T}; \pi_d^\phi)), \theta\big)\big| \le g_\phi(\omega), \quad \text{and} \quad \int_\Omega g_\phi(\theta, \varepsilon_{1:T})\, \mathrm{d}\mu(\theta, \varepsilon_{1:T}) < \infty, \tag{27}$$
and
$$\big|\partial_\psi\, \ell\big(\pi_a^\psi(h_T(\theta, \varepsilon_{1:T}; \pi_d^\phi)), \theta\big)\big| \le g_\psi(\omega), \quad \text{and} \quad \int_\Omega g_\psi(\theta, \varepsilon_{1:T})\, \mathrm{d}\mu(\theta, \varepsilon_{1:T}) < \infty. \tag{28}$$

These conditions follow from the differentiation-under-the-integral theorem. The following section explains how these sufficient conditions for differentiating under the integral sign can be derived from assumptions on the model.

C.3 Conditions on the BED model

Let $\Phi \subset \mathbb{R}^{d_\phi}$ and $\Psi \subset \mathbb{R}^{d_\psi}$ be open parameter sets. In order to have the map
$$(\phi, \psi) \mapsto \ell(\pi_a^\psi(h_T(\theta, \varepsilon_{1:T}; \pi_d^\phi)), \theta)$$
differentiable, we have to satisfy the following conditions:

1. **Differentiable design network**: The design-induced history
$$h_T : \Theta \times \mathcal{E} \times \Phi \to \mathcal{H}_T$$
$$(\theta, \varepsilon_{1:T}, \phi) \mapsto h_T(\theta, \varepsilon_{1:T}; \pi_d^\phi),$$
is measurable in $(\theta, \varepsilon_{1:T})$ and differentiable in ϕ for almost all $(\theta, \varepsilon_{1:T})$, with
$$\nabla_\phi h_T(\theta, \varepsilon_{1:T}, \phi)$$
existing and being finite. In particular, in all deep learning architectures considered to model h_T, this is implemented as a composition of differentiable operations with respect to ϕ (e.g. affine maps and smooth or piecewise-smooth activations), so that it is differentiable almost everywhere.
2. **Differentiable prediction network**: The downstream prediction network
$$\pi_a^\psi : \mathcal{H}_T \to \mathcal{A}$$
is measurable in its input and differentiable in (ψ, h_T), i.e.
$$(\psi, h) \mapsto \pi_a^\psi(h)$$
is differentiable for all (ψ, h), or at least almost everywhere, with Jacobians
$$\nabla_\psi \pi_a^\psi(h), \qquad \nabla_h \pi_a^\psi(h)$$
existing and finite. This holds in particular when π_a^ψ is a neural network with standard smooth (or piecewise-linear) activations such as $\tanh$, $\mathrm{softplus}$, ReLU or $\mathrm{leaky-ReLU}$, which are differentiable almost everywhere.
3. **Differentiable loss**: The downstream loss
$$\ell : \mathcal{A} \times \Theta \to \mathbb{R}, \quad (a, \theta) \mapsto \ell(a, \theta),$$
is differentiable in its first argument a for all (a, θ) (or almost everywhere), with gradient
$$\nabla_a \ell(a, \theta)$$
existing and finite (e.g. mean-squared error, cross-entropy, etc.).

Under these conditions, for almost all $(\theta, \varepsilon_{1:T})$, the composite map
$$(\phi, \psi) \longmapsto \ell\big(\pi_a^\psi(h_T(\theta, \varepsilon_{1:T}, \phi)), \theta\big)$$
is differentiable, and its gradient can be computed by the chain rule.

D Gradient Computations

D.1 Gradient Computations for Separated Networks

We denote, for a single draw $(\theta, \varepsilon_{1:T}) \sim p(\theta)q(\varepsilon_{1:T})$,

$$L(\phi, \psi; \theta, \varepsilon_{1:T}) := \ell\big(\pi_a^\psi(h_T(\theta, \varepsilon_{1:T}; \pi_d^\phi)),\, \theta\big),$$

so that

$$\widehat{\nabla_{\phi,\psi}}\,\mathrm{EFL}(\phi, \psi) = \nabla_{\phi,\psi} L(\phi, \psi; \theta, \varepsilon_{1:T}). \tag{29}$$

Gradient with respect to ψ. Since h_T does not depend on ψ, we have

$$\nabla_\psi L(\phi, \psi; \theta, \varepsilon_{1:T}) = \nabla_\psi\, \ell\big(\pi_a^\psi(h_T(\theta, \varepsilon_{1:T}; \pi_d^\phi)),\, \theta\big) \tag{30}$$

$$= \nabla_a \ell\big(a, \theta\big)\Big|_{a=\pi_a^\psi(h_T)}\, \nabla_\psi \pi_a^\psi\big(h_T(\theta, \varepsilon_{1:T}; \pi_d^\phi)\big), \tag{31}$$

where $\nabla_a \ell(a, \theta)$ denotes the gradient of the loss with respect to its first argument.

Gradient with respect to ϕ. Here the dependence on ϕ is only through the history $h_T(\theta, \varepsilon_{1:T}; \pi_d^\phi)$, so the chain rule yields

$$\nabla_\phi L(\phi, \psi; \theta, \varepsilon_{1:T}) = \nabla_\phi\, \ell\big(\pi_a^\psi(h_T(\theta, \varepsilon_{1:T}; \pi_d^\phi)),\, \theta\big) \tag{32}$$

$$= \nabla_a \ell\big(a, \theta\big)\Big|_{a=\pi_a^\psi(h_T)}\, \nabla_{h_T} \pi_a^\psi\big(h_T(\theta, \varepsilon_{1:T}; \pi_d^\phi)\big)\, \nabla_\phi h_T(\theta, \varepsilon_{1:T}; \pi_d^\phi). \tag{33}$$

Joint gradient estimator. Stacking the two components, a single-sample Monte Carlo estimator of the full gradient is thus

$$\widehat{\nabla_{\phi,\psi}}\,\mathrm{EFL}(\phi, \psi) = \begin{pmatrix} \nabla_\phi L(\phi, \psi; \theta, \varepsilon_{1:T}) \\ \nabla_\psi L(\phi, \psi; \theta, \varepsilon_{1:T}) \end{pmatrix}.$$

In practice, automatic differentiation computes precisely these quantities by backpropagating through the computational graph

$$(\phi, \psi, \theta, \varepsilon_{1:T}) \mapsto h_T(\theta, \varepsilon_{1:T}; \pi_d^\phi) \mapsto \pi_a^\psi(h_T) \mapsto \ell(\pi_a^\psi(h_T), \theta).$$

D.2 Gradient Estimator in case of weight sharing

We now make explicit the form of the Monte Carlo gradient estimator in the case where both the acquisition and downstream policies share the same encoder network, with a sum-pooling aggregation step of the encodings and a specific prediction head for both. It is a common choice of architecture for policy-based methods. For a single draw $(\theta, \varepsilon_{1:T}) \sim p(\theta)q(\varepsilon_{1:T})$, we consider the loss

$$L(\phi_1, \phi_2, \psi; \theta, \varepsilon_{1:T}) := \ell\big(\pi_a^\psi(h_T(\theta, \varepsilon_{1:T}; \pi_d^{\phi_1,\phi_2})),\, \theta\big),$$

where the downstream prediction network is defined via a shared DeepSets-style encoder:

$$e_t = E_{\phi_1}(\xi_t, y_t) \in \mathbb{R}^d, \qquad t = 1, \dots, T, \tag{34}$$

$$R(h_T) = \sum_{t=1}^T e_t = \sum_{t=1}^T E_{\phi_1}(\xi_t, y_t), \tag{35}$$

$$\pi_a^\psi(h_T) = f_\psi\big(R(h_T)\big). \tag{36}$$

A single-sample Monte Carlo estimator of the gradient of the expected future loss is then

$$\widehat{\nabla_{\phi_1,\phi_2,\psi}}\,\mathrm{EFL}(\phi_1, \phi_2, \psi) = \nabla_{\phi_1,\phi_2,\psi}\, L(\phi_1, \phi_2, \psi; \theta, \varepsilon_{1:T}).$$

Gradient with respect to the prediction head ψ. The dependence on ψ is only through $a = f_\psi(R(h_T))$, so the chain rule yields

$$\nabla_\psi L(\phi_1, \phi_2, \psi; \theta, \varepsilon_{1:T}) = \nabla_\psi\, \ell\big(f_\psi(R(h_T)), \theta\big) \tag{37}$$

$$= \nabla_a \ell\big(a, \theta\big)\Big|_{a=f_\psi(R(h_T))}\, \nabla_\psi f_\psi\big(R(h_T)\big). \tag{38}$$

Gradient with respect to the shared encoder parameters ϕ_1. Here, ϕ_1 affects the loss both through the encoder outputs $e_t = E_{\phi_1}(\xi_t, y_t)$ and, through the design policy, through the generated history $h_T(\theta, \varepsilon_{1:T}; \pi_d^{\phi_1,\phi_2})$. Applying the chain rule gives

$$\nabla_{\phi_1} L(\phi_1, \phi_2, \psi; \theta, \varepsilon_{1:T}) = \nabla_a \ell\big(a, \theta\big)\Big|_{a=f_\psi(R(h_T))} \nabla_R f_\psi\big(R(h_T)\big)\ \nabla_{\phi_1} R(h_T). \tag{39}$$

The term $\nabla_{\phi_1} R(h_T)$ includes both the direct dependence of the encoder E_{ϕ_1} on ϕ_1 and the indirect dependence through the history $h_T(\theta, \varepsilon_{1:T}; \pi_d^{\phi_1,\phi_2})$. In particular,

$$\nabla_{\phi_1} R(h_T) = \nabla_{\phi_1} \left[\sum_{t=1}^{T} E_{\phi_1}(\xi_t, y_t)\right], \tag{40}$$

where $(\xi_t, y_t)_{t=1}^T$ are generated by the design policy $\pi_d^{\phi_1,\phi_2}$.

Gradient with respect to the design head ϕ_2. The parameters ϕ_2 enter through the design policy that generates the history $h_T(\theta, \varepsilon_{1:T}; \pi_d^{\phi_1,\phi_2})$. Denoting this dependence explicitly, the chain rule gives

$$\nabla_{\phi_2} L(\phi_1, \phi_2, \psi; \theta, \varepsilon_{1:T}) = \nabla_{h_T}\, \ell\big(f_\psi(R(h_T)), \theta\big)\ \nabla_{\phi_2} h_T(\theta, \varepsilon_{1:T}; \pi_d^{\phi_1,\phi_2}), \tag{41}$$

where the term $\nabla_{h_T} \ell(\cdot)$ itself decomposes through the encoder and pooling:

$$\nabla_{h_T}\, \ell\big(f_\psi(R(h_T)), \theta\big) = \nabla_a \ell\big(a, \theta\big)\Big|_{a=f_\psi(R(h_T))} \nabla_R f_\psi\big(R(h_T)\big)\ \nabla_{h_T} R(h_T). \tag{42}$$

In practice, automatic differentiation computes exactly these gradients by backpropagating through the computational graph

$$(\phi_1, \phi_2, \psi, \theta, \varepsilon_{1:T}) \mapsto h_T(\theta, \varepsilon_{1:T}; \pi_d^{\phi_1,\phi_2}) \mapsto \{E_{\phi_1}(\xi_t, y_t)\}_{t=1}^T \mapsto R(h_T) \mapsto f_\psi(R(h_T)) \mapsto \ell(\cdot, \theta).$$

Finally, a Monte Carlo estimator of the full gradient is obtained by averaging these single-sample gradients over i.i.d. draws $(\theta^{(i)}, \varepsilon_{1:T}^{(i)})$.

E Network Architecture

Our co-optimization framework, introduced in Section 4, is agnostic to the specific parameterization of the design policy π_d^ϕ and the downstream action policy π_a^ψ. Any differentiable architecture can be used, provided that the resulting computation graph supports pathwise optimization through the reparameterized simulator. Nevertheless, architectural choices play an important role in practice, since they determine how the experimental history is represented and which inductive biases are made available to the policy. In particular, the architecture can encode exchangeability, temporal ordering, Markovian structure, spatial symmetries, or more general dependencies between past designs and outcomes.

Design policy. The design policy π_d^ϕ is the central component of the acquisition mechanism. At each experimental step, it maps the current history

$$h_t = \{(\xi_s, y_s)\}_{s=1}^t$$

to the next design, or to a distribution over the designs,

$$\begin{cases} \xi_{t+1} = \pi_d^\phi(h_t) \\ \xi_{t+1} \sim p_\phi(\xi \mid h_t) =: \pi_d^\phi(h_t) \end{cases}$$

In deep adaptive design, this mapping is commonly parameterized through an encoder–aggregator–emitter architecture. First, each observed design–outcome pair (ξ_s, y_s) is embedded by a local encoder,

$$r_s = E_{\phi_1}(\xi_s, y_s),$$

or, more generally, by separate design and outcome encoders followed by a fusion module. The resulting sequence or set of pair embeddings $(r_1, \dots, r_t)$ is then compressed into a global history representation,

$$z_t = A_{\phi_2}(r_1, \dots, r_t),$$

where A_{ϕ_2} is a history aggregation module. Finally, an emitter network maps this history state to the next design,

$$\xi_{t+1} = G_{\phi_3}(z_t),$$

or to a distribution over candidate designs in stochastic acquisition policies.

The main decision-making capacity of the design policy is often concentrated in the aggregation module. The encoder produces local representations of individual experiments, whereas the aggregator determines how these pieces of evidence are combined to support the next acquisition decision. Different choices of A_{ϕ_2} encode different assumptions about the structure of the history. For exchangeable probabilistic models of histories, permutation-invariant architectures such as DeepSets [24, 74] are natural: they aggregate pair embeddings through a symmetric pooling operation and therefore make the design decision independent of the ordering of observations. For temporally ordered or dynamically evolving systems, recurrent architectures such as LSTMs [31, 42] can encode the sequential dependence between past interventions and outcomes. Attention-based architectures, including Set Transformers [51] and Transformer Neural Process (TNP)-style encoders [3], provide a more flexible alternative by allowing the policy to learn interactions between all past design–outcome pairs and to condition the next design on the most informative parts of the history.

Downstream action network. The downstream action policy maps the final experimental history to a terminal action,

$$a_T = \pi_a^\psi(h_T).$$

This action may represent a point estimate, a classifier output, a decision rule, or any task-specific prediction required after data acquisition. Since the downstream policy is only evaluated after the acquisition phase and needs to extract information contained in the history, π_a^ψ can be implemented with a lighter architecture, or can follow the same encoder–aggregator–emitter structure as the design policy:

$$r_s = E_{\psi_1}(\xi_s, y_s), \qquad z_T = A_{\psi_2}(r_1, \dots, r_T), \qquad a_T = G_{\psi_3}(z_T).$$

This shared architectural template is especially useful when the downstream action depends on subtle interactions across the entire history. In such cases, using a rich history encoder for π_a^ψ can improve the quality of the terminal decision and provide stronger gradients to guide the design policy during co-optimization.

Enhancing coupling through weight sharing. When the design policy and downstream action policy are built from compatible encoder–aggregator architectures, one can further strengthen their coupling by sharing part of their representations. For example, both policies may use a common local encoder E_ω for design–outcome pairs, while retaining separate aggregation and output heads:

$$r_s = E_\omega(\xi_s, y_s),$$

$$\xi_{t+1} = G_d^\phi\left(A_d^\phi(r_1, \dots, r_t)\right), \qquad a_T = G_a^\psi\left(A_a^\psi(r_1, \dots, r_T)\right).$$

A stronger form of sharing can also be used, where the local encoder and part of the history aggregator are shared, while task-specific heads remain distinct:

$$z_t = A_\omega(E_\omega(\xi_1, y_1), \dots, E_\omega(\xi_t, y_t)),$$

$$\xi_{t+1} = G_d^\phi(z_t), \qquad a_T = G_a^\psi(z_T).$$

This form of weight sharing has several advantages. First, it encourages both the design policy and the downstream predictor to build compatible representations of the experimental history. The design policy is then trained to collect data that are useful under the same representational geometry used by the downstream action network. Second, sharing parameters strengthens the coupling between the two objectives: gradients from the downstream loss directly shape the representation used for acquisition. Third, it reduces the number of free parameters, which can improve statistical efficiency and regularize training, especially when the number of simulated trajectories is limited.

E.1 TNP-Like Policy Architecture

In the experiments, we use a modular neural architecture for the adaptive design policy π_d^ϕ. The architecture follows the general encoder–aggregator–emitter decomposition described above, with a Transformer-based module used to summarize the experimental history. Its purpose is to map the information collected so far into the next design decision, while remaining flexible enough to handle different experimental tasks. The architecture can be viewed as a TNP-like design policy where past design–outcome pairs play the role of context observations, and a learned decision token queries this context in order to produce the next design. We detail below the different modules composing the architecture used throughout the experiments. Experiment-specific dimensions and hyperparameters are denoted by variables, whereas numerical values indicate settings shared across all experiments.

Encoding design–outcome pairs. Each past design ξ_t and outcome y_t is first mapped to a latent representation. In practice, the design and outcome are encoded separately,

$$e_t^\xi = f_\xi(\xi_t), \qquad e_t^y = f_y(y_t),$$

Table 5: Design encoder architecture.

Layer	Description	Dimension	Activation
Input	ξ	$\dim_\xi$	–
H1	Fully connected	64	ReLU
Output	Fully connected	$d_{\text{enc},\xi}$	–

Table 6: Outcome encoder architecture.

Layer	Description	Dimension	Activation
Input	y	$\dim_y$	–
H1	Fully connected	64	ReLU
Output	Fully connected	$d_{\text{enc},y}$	–

and then combined into a single representation of the experiment performed at step t:

$$r_t = f_{\text{pair}}\left([e_t^\xi; e_t^y]\right).$$

Table 7: Fusion module architecture.

Layer	Description	Dimension	Activation
Input	$[e^\xi; e^y]$	$d_{\text{enc},y} + d_{\text{enc},\xi}$	–
H1	Fully connected	64	GELU
Output	Fully connected	$d_{\text{enc},r}$	–

This pair representation captures the local information revealed by one design–outcome interaction. Applying the same construction to all previous steps produces a sequence of history tokens

$$R_t = (r_1, \dots, r_t).$$

When the task is temporally structured, these tokens can also be augmented with time information, so that the policy can distinguish early and late observations. When the task is exchangeable, the same representation can instead be interpreted as an unordered set of observed experiments. In all experiments reported in this work, we augment these tokens with learned time embeddings, allowing the policy to distinguish observations according to their position in the acquisition trajectory.

Aggregating the experimental history. The sequence of pairwise representations $(r_1, \dots, r_t)$ is processed by a Transformer-based history aggregator. Each representation is first projected to a common embedding dimension, then enriched with a learned type embedding indicating that it corresponds to a history token. A learned temporal embedding is also added to preserve the order of the acquisition trajectory. A learned decision token is appended at the end of the sequence and is assigned its own type embedding, together with the time index corresponding to the next decision step. The resulting token sequence is normalized and passed through a stack of Transformer encoder layers.

The final representation of the decision token is used as the summary of the experimental history. This token acts as a learnable query that attends to all previous design–outcome pairs and extracts the information relevant for choosing the next design. Denoting this representation by h_t, the aggregation step can be written as

$$h_t = f_{\text{hist}}(r_1, \dots, r_t), \qquad h_t \in \mathbb{R}^{d_h}.$$

This attention-based aggregation avoids imposing a fixed pooling rule over the experimental history. It allows the policy to model interactions between past observations, account for the temporal order of the acquisitions, and focus on the parts of the trajectory that are most informative for the next experimental decision.

Table 8: History aggregator architecture. The module maps the sequence of pair representations $(r_1, \dots, r_t)$ to a single history representation h_t.

Component	Configuration	Output shape
Input sequence	Pair representations $(r_1, \dots, r_t)$, $r_i \in \mathbb{R}^{d_{\text{enc},r}}$	$t \times d_{\text{enc},r}$
Input adapter	Linear layer $d_{\text{enc},r} \to d_{\text{enc},h}$	$t \times d_{\text{enc},h}$
History token embeddings	Add learned history type embeddings	$t \times d_{\text{enc},h}$
Time embeddings	Add learned time embeddings, enabled	$t \times d_{\text{enc},h}$
Decision token	Append learned query token with decision type embedding	$(t+1) \times d_{\text{enc},h}$
Transformer encoder	2 layers, 4 heads, feedforward dimension 128	$(t+1) \times d_{\text{enc},h}$
Readout	Final decision-token representation	$d_{\text{enc},h}$

Emitting the next design. Finally, an emitter network maps the history representation to the next design:

$$\xi_{t+1} = f_{\text{emit}}(h_t).$$

Table 9: Emitter network architecture.

Layer	Description	Dimension	Activation
Input	History representation h_t	$d_{\text{enc},h}$	–
H1	Fully connected	64	GELU
Output	Fully connected	$\dim_\xi$	–

Complete and modular design policy. The complete design policy is obtained by composing the modules. Given the experimental history $\mathcal{H}_t = \{(\xi_i, y_i)\}_{i=1}^t$, the policy can be written as

$$\pi_\phi(\mathcal{H}_t) = f_{\text{emit}} \circ f_{\text{hist}} \circ f_{\text{pair}} \circ (f_\xi, f_y)(\{(\xi_i, y_i)\}_{i=1}^t).$$

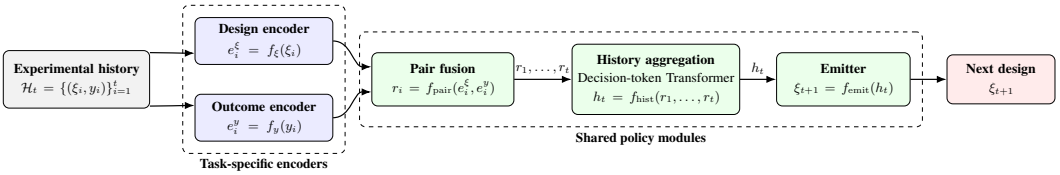


Figure 3: Modular architecture of the design policy. Past designs and outcomes are encoded separately, fused into pair representations, aggregated by a decision-token Transformer, and mapped by the emitter to the next design.

A key advantage of this architecture is its modularity. The task-specific encoders f_ξ and f_y can be adapted to the structure of each experiment, while the fusion, history aggregation, and emission modules are kept fixed. In low-dimensional settings, the encoders are simple multilayer perceptrons, whereas for structured or high-dimensional observations, f_y can be replaced by a convolutional or patch-based encoder. This architecture therefore provides a common backbone across all experiments, both for our task-driven method and for the DAD/iDAD baselines. The same policy network can be trained end-to-end by backpropagating gradients either through the downstream task loss or through the sequential PCE objective.

E.2 Downstream warm-up

In practice, before running the joint optimisation described in algorithm 1, we may initialise the action policy with a short downstream warm-up phase. During this phase, the design policy is kept fixed and is not used to generate designs. Instead, design trajectories are sampled from a task-relevant random distribution $r(\xi_{1:T})$, and only the downstream-action policy π_a^ψ is trained to minimise the target downstream loss. Given samples $\theta^b \sim p(\theta)$, simulator noise $\varepsilon_{1:T}^b \sim q(\varepsilon_{1:T})$, and random designs $\xi_{1:T}^b \sim r(\xi_{1:T})$, the warm-up gradient estimator is

$$\widehat{\nabla_\psi}\, \mathcal{L}_{\text{down}}(\psi) = \frac{1}{B} \sum_{b=1}^{B} \nabla_\psi \ell\big(\pi_a^\psi(h_T(\theta^b, \varepsilon_{1:T}^b; \xi_{1:T}^b)), \theta^b\big). \tag{43}$$

Although the resulting random designs are generally suboptimal, they provide an initial supervised signal for the action policy. Empirically, this warm-up can improve the subsequent joint optimisation: when the design policy starts being trained, the downstream network already maps histories to meaningful actions, yielding more informative gradients for the design policy and leading to a better coupling between acquisition and downstream prediction.

Algorithm 2 ACTION-BED with downstream warm-up

Require: Policies π_d^ϕ, π_a^ψ, initialisations ϕ_0, ψ_0
Require: Warm-up steps K_{warm}, joint steps K, random design distribution $r(\xi_{1:T})$
Downstream warm-up
for $k = 0, 1, \ldots, K_{\text{warm}} - 1$ **do**
  $\theta^{1:B}, \varepsilon_{1:T}^{1:B}, \xi_{1:T}^{1:B} \sim p(\theta) q(\varepsilon_{1:T}) r(\xi_{1:T})$
  $g_k^\psi \leftarrow \widehat{\nabla_\psi}\, \mathcal{L}_{\text{down}}(\psi_k)$ (eq. 43)
  $\psi_{k+1} \leftarrow \text{UPDATE}_a(\psi_k, g_k^\psi)$
end for
$\psi_0 \leftarrow \psi_{K_{\text{warm}}}$
Joint Action-BED optimisation
for $k = 0, 1, \ldots, K - 1$ **do**
  $\theta^{1:B}, \varepsilon_{1:T}^{1:B} \sim p(\theta) q(\varepsilon_{1:T})$
  $g_k^\phi, g_k^\psi \leftarrow \widehat{\nabla_{\phi,\psi}}\, \text{EFL}(\phi_k, \psi_k)$ (eq. (10))
  $\phi_{k+1}, \psi_{k+1} \leftarrow \text{UPDATE}(\phi_k, g_k^\phi, \psi_k, g_k^\psi)$
end for
return ϕ_K, ψ_K

F Bayes-optimal action under MSE and permutation-invariant MSE

We provide here a self-contained derivation of the Bayes-optimal action associated with the downstream decision problem considered in the source location finding experiment in Section 6.1. Since the general Bayes-action formulation is introduced in the main text, we only recall the minimal setup needed for the derivation. We then derive the posterior-mean optimality result for the standard squared error loss and extend it to the permutation-invariant MSE used in the experiment.

F.1 Bayes-action setup

Let $\theta \in \Theta$ denote the latent parameter and let $h_T \in \mathcal{H}_T$ denote the experimental trajectory induced by a fixed data-acquisition policy π_d^ϕ. A downstream action policy is a measurable function

$$\pi_a^\psi : \mathcal{H}_T \to \mathcal{A},$$

implemented in practice by the downstream-action network.

For a loss function $\ell : \mathcal{A} \times \Theta \to \mathbb{R}_+$, and a fixed trajectory h_T, we call the Bayes-optimal action the minimizer of the posterior expected loss:

$$a^\star(h_T) = \arg\min_{a \in \mathcal{A}} \mathbb{E}_{p(\theta|h_T)} \left[\ell(a, \theta)\right]. \tag{44}$$

F.2 Standard MSE loss

We first specialize to the standard squared error loss. We assume that $\Theta = \mathcal{A} \subseteq \mathbb{R}^d$, which is the case for the source location finding task, and we define

$$\ell_{\mathrm{MSE}}(\hat{\theta}, \theta) = \|\hat{\theta} - \theta\|_2^2.$$

For a fixed trajectory h_T, the conditional risk is

$$\mathcal{R}_{\mathrm{MSE}}(\hat{\theta} \mid h_T) = \mathbb{E}_{p(\theta|h_T)} \left[\|\hat{\theta} - \theta\|_2^2\right]. \tag{45}$$

Expanding the squared norm gives

$$\begin{aligned} \mathcal{R}_{\mathrm{MSE}}(\hat{\theta} \mid h_T) &= \mathbb{E}\left[\|\hat{\theta}\|_2^2 - 2\langle\hat{\theta}, \theta\rangle + \|\theta\|_2^2 \,\middle|\, h_T\right] \\ &= \|\hat{\theta}\|_2^2 - 2\left\langle\hat{\theta}, \mathbb{E}[\theta \mid h_T]\right\rangle + \mathbb{E}\left[\|\theta\|_2^2 \mid h_T\right]. \end{aligned} \tag{46}$$

The last term does not depend on $\hat{\theta}$. Differentiating with respect to $\hat{\theta}$ and setting the gradient to zero yields

$$\nabla_{\hat{\theta}} \mathcal{R}_{\mathrm{MSE}}(\hat{\theta} \mid h_T) = 2\hat{\theta} - 2\mathbb{E}[\theta \mid h_T] = 0.$$

Hence,

$$\hat{\theta}^\star_{\mathrm{MSE}}(h_T) = \mathbb{E}[\theta \mid h_T]. \tag{47}$$

Thus, under the standard MSE loss, the Bayes-optimal action is the posterior mean.

F.3 Permutation-invariant MSE loss

However, in our setting the latent parameter is not naturally ordered. For example, in the source location finding problem with K exchangeable sources, the parameter can be written as

$$\theta = (\theta_1, \ldots, \theta_K) \in \mathbb{R}^{K \times d},$$

but the ordering of the sources is arbitrary. In that case, penalizing an estimate $\hat{\theta}$ using the standard MSE would incorrectly distinguish between two estimates that differ only by a permutation of the sources. Let $\mathcal{S}_K$ denote the symmetric group over K elements. For a permutation $\sigma \in \mathcal{S}_K$, write

$$\sigma \cdot \theta = (\theta_{\sigma(1)}, \ldots, \theta_{\sigma(K)}).$$

The permutation-invariant MSE is defined as

$$\ell_{\text{PI-MSE}}(\hat{\theta}, \theta) = \min_{\sigma \in \mathcal{S}_K} \frac{1}{Kd} \sum_{k=1}^{K} \left\|\hat{\theta}_k - \theta_{\sigma(k)}\right\|_2^2. \tag{48}$$

Equivalently, identifying permutations with permutation matrices $P \in \mathcal{P}_K$, this can be written as

$$\ell_{\text{PI-MSE}}(\hat{\theta}, \theta) = \frac{1}{Kd} \min_{P \in \mathcal{P}_K} \left\| \hat{\theta} - P\theta \right\|_F^2, \tag{49}$$

where $\| \cdot \|_F$ denotes the Frobenius norm.

For a fixed trajectory h_T, the Bayes-optimal action under this loss is therefore

$$\hat{\theta}^\star_{\text{PI-MSE}}(h_T) = \arg \min_{\hat{\theta} \in \mathbb{R}^{K \times d}} \mathbb{E}_{p(\theta | h_T)} \left[\frac{1}{Kd} \min_{P \in \mathcal{P}_K} \left\| \hat{\theta} - P\theta \right\|_F^2 \right]. \tag{50}$$

This is the Fréchet mean of the posterior distribution on the quotient space $\mathbb{R}^{K \times d} / \mathcal{S}_K$, where configurations that differ only by a permutation are identified. To make the optimality condition explicit, define, for a given estimate $\hat{\theta}$ and latent parameter θ, an optimal alignment

$$P_{\hat{\theta}}(\theta) \in \arg \min_{P \in \mathcal{P}_K} \left\| \hat{\theta} - P\theta \right\|_F^2. \tag{51}$$

Assuming that this optimal alignment is unique almost surely under $p(\theta \mid h_T)$, the conditional risk can be written locally as

$$\mathcal{R}_{\text{PI-MSE}}(\hat{\theta} \mid h_T) = \frac{1}{Kd} \cdot \mathbb{E}_{p(\theta | h_T)} \left[\left\| \hat{\theta} - P_{\hat{\theta}}(\theta)\theta \right\|_F^2 \right]. \tag{52}$$

By the envelope theorem, differentiating the minimized objective with respect to $\hat{\theta}$ gives

$$\begin{aligned} \nabla_{\hat{\theta}} \mathcal{R}_{\text{PI-MSE}}(\hat{\theta} \mid h_T) &= \frac{2}{Kd} \cdot \mathbb{E}_{p(\theta | h_T)} \left[\hat{\theta} - P_{\hat{\theta}}(\theta)\theta \right] \\ &= \frac{2}{Kd} \cdot \left(\hat{\theta} - \mathbb{E}_{p(\theta | h_T)} \left[P_{\hat{\theta}}(\theta)\theta \right] \right). \end{aligned} \tag{53}$$

Therefore, any differentiable Bayes-optimal action satisfies the fixed-point condition

$$\hat{\theta}^\star_{\text{PI-MSE}}(h_T) = \mathbb{E}_{p(\theta | h_T)} \left[P_{\hat{\theta}^\star_{\text{PI-MSE}}(h_T)}(\theta)\theta \right]. \tag{54}$$

Equation (54) shows that the Bayes-optimal action under permutation-invariant MSE is the posterior mean after optimally aligning each posterior sample to the estimate itself. The Bayes-optimal action is also not unique in the usual Euclidean sense: if $\hat{\theta}^\star$ is optimal, then $P\hat{\theta}^\star$ is also optimal for any permutation matrix $P \in \mathcal{P}_K$. Hence, the estimator is identifiable only up to permutation, which is precisely the desired invariance property.

F.4 Interpretation for the downstream network

Under the standard MSE loss, minimizing the empirical Bayesian risk trains the downstream-action network to approximate the posterior mean mapping

$$h_T \longmapsto \mathbb{E}[\theta \mid h_T].$$

Under the permutation-invariant MSE, the downstream network instead learns a representative of the posterior Fréchet mean on the quotient space induced by the permutation symmetry:

$$h_T \longmapsto \arg\min_{\hat{\theta}} \mathbb{E}_{p(\theta | h_T)} \left[\min_{P \in \mathcal{P}_K} \frac{1}{Kd} \left\| \hat{\theta} - P\theta \right\|_F^2 \right].$$

Equivalently, it learns an unordered set-valued estimate of the latent sources. During training and evaluation, the optimal matching in Equation (48) removes the arbitrary label ordering of the sources, ensuring that two estimates that differ only by a permutation incur the same loss.

G Experiment Details

The implementation was carried out in part using the PyTorch library [60], and the code is publicly available. Experiments were conducted on a server equipped with an NVIDIA Titan RTX GPU (24GB, Turing architecture) and 24 CPU cores. Each experiment was run on a single GPU.

G.1 Source Location Finding

The *source location finding* problem is a canonical benchmark in BED, in which the objective is to infer the unknown spatial positions of K signal-emitting sources $\theta_1, \dots, \theta_K \in \mathbb{R}^2$ from a sequence of noisy measurements. Following the experimental setting of [24], at each design step t, an agent selects a sensor location $\xi_t \in \mathbb{R}^2$, and observes a scalar response $y_t \in \mathbb{R}$ corresponding to the aggregation of the signals emitted by the K sources, whose expected magnitudes decay with the squared distance to the sensor according to a known physical attenuation model. A commonly used formulation assumes

$$\log(y_t) = \log\left(b + \sum_{k=1}^{K} \frac{A}{m + \|\xi_t - \theta_k\|^2}\right) + \sigma\,\varepsilon_t, \qquad \varepsilon_t \sim \mathcal{N}(0,1), \tag{55}$$

where $A > 0$ denotes the signal amplitude, b is a background offset, $m > 0$ is a stabilising constant preventing singularities at zero distance, and σ controls the measurement noise level. Equivalently, denoting the noiseless aggregated intensity by

$$\mu(\theta, \xi) = b + \sum_{k=1}^{K} \frac{A}{m + \|\xi - \theta_k\|^2},$$

the observation model can be written in terms of noisy measurements of the log total intensity, yielding the prior–likelihood specification

$$\theta_k \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0_2, I_2), \qquad \log y \mid \theta, \xi \sim \mathcal{N}\big(\log \mu(\theta, \xi), \sigma^2\big),$$

where $\theta = (\theta_1, \dots, \theta_K) \in \mathbb{R}^{K\times 2}$. This formulation clearly shows that the model satisfies the reparameterization assumption C.1. The goal is to localize the sources with the highest possible accuracy: we seek to collect data from which a downstream model can be trained to identify all sources as effectively as possible. In our experimental setting, we consider the case $K = 2$.

The task is challenging because information is spatially structured and highly non-uniform: sensor placements far from the source yield almost uninformative responses, while overly greedy exploitation may lead to premature localisation around suboptimal regions. From an inference perspective, the posterior distribution $p(\theta \mid y_{1:T}, \xi_{1:T})$ is typically multi-modal at early stages and becomes sharply concentrated as informative measurements accumulate. Thus, in this setting, amortised methods such as DAD aim to query maximally informative sensors and should generate sequences of designs that are effective for the prediction task.

Table 10: **Location Finding Parameters**

Parameter	Value
Number of sources, K	2
$\dim_\xi$	2
$\dim_y$	1
T	30
Base signal, b	10^{-1}
Max. signal, m	10^{-4}
A	1
Signal noise, σ	0.5

Downstream Objective. In this experimental setting, the downstream objective is to predict the source locations. Prediction quality is evaluated using two losses: the *Mean Squared Error* (MSE) and a logarithmic variant, denoted *Quadratic-Logarithmic Loss* (QLL). For problems with multiple sources, the source labels become exchangeable, meaning that permuting their ordering leaves both

the likelihood and the signal observed by the sensor unchanged. Hence, the downstream task is intrinsically permutation-invariant. What matters is the predicted set of source locations, not the arbitrary ordering assigned to its elements. Consequently, the standard MSE loss is not the most relevant evaluation criterion in this setting, since it imposes a fixed correspondence between predicted and true sources and may penalize predictions that are correct up to a permutation. Therefore, we use in this experiment the permutation invariant MSE over the losses, defined as

$$\ell_{\mathrm{MSE}}(\theta, \hat{\theta}) = \min_{\sigma \in \mathcal{S}_K} \frac{1}{2K} \cdot \sum_{k=1}^{K} \|\hat{\theta}_{\sigma(k)} - \theta_k\|^2, \tag{56}$$

where $\mathcal{S}_K$ is the set of permutations of $\{1, \cdots, K\}$ and $\hat{\theta}$ the point prediction of source locations. This optimization problem is significantly more complex than in the single-source case. The expectation involves a minimization over permutations inside the loss, leading to a highly non-linear and generally non-convex objective. This phenomenon poses significant challenges during training:

- **Label switching and multimodality.** The posterior distribution is invariant under permutations of the sources, which typically results in a multimodal posterior with $K!$ symmetric modes.
- **Failure of the posterior mean.** Averaging over these modes leads to a posterior mean that does not correspond to any meaningful configuration of sources, often placing predicted sources between true locations or collapsing them toward the center of mass.

As a result, under this permutation-invariant loss, the Bayes estimator is not the usual posterior mean on the labelled parameter space, but rather the posterior Fréchet mean induced by the optimal-assignment metric. Equivalently, the Bayes-optimal action is a fixed point where each predicted source is equal to the posterior mean of the true sources optimally matched to it; see Appendix F.3. This loss is adopted because it directly reflects the objective of the experiment, namely recovering the unordered set of source locations. For the *Quadratic-Logarithmic Loss*, we similarly account for the permutation symmetry of the source configuration by minimizing over all possible assignments between the predicted and true source locations. Specifically, we define the permutation-invariant QLL as

$$\ell_{\mathrm{QLL}}(\theta, \hat{\theta}) = \min_{\sigma \in \mathcal{S}_K} \frac{1}{KD} \sum_{k=1}^{K} \sum_{d=1}^{D} \left[\left(\theta_{\sigma(k),d} - \hat{\theta}_{k,d} \right)^2 + \log\left(\left(\theta_{\sigma(k),d} - \hat{\theta}_{k,d} \right)^2 + \epsilon \right) \right] \tag{57}$$

where $\epsilon = 10^{-6}$ is used for numerical stability. By selecting the permutation that minimizes the loss, this formulation accounts for the fact that source identities are interchangeable, without imposing an arbitrary ordering on the source locations. Consequently, the downstream network is trained to predict the unordered source configuration from the observed trajectory $\{(\xi_i, y_i)\}_{i=1}^{T}$.

Experimental Details. In this experiment, we consider three baseline methods:

1. **RANDOM**, where designs are naively sampled at random from the marginal prior distribution of a single source location.
2. **DAD**, the standard deep policy-based method introduced in [24]. It parameterises the adaptive design policy with a neural network and is trained using its native expected-information-gain objective. To ensure a fair comparison, we replace the architectural backbone of the original implementation with the same policy network used in our method.
3. **ALINE**, a recent policy-based method that achieves among the strongest reported performance in the literature. Its effectiveness is driven by a TNP-like policy architecture, a dense trajectory-level reward objective, and an internal posterior-approximation mechanism that amortises posterior inference. We retain the architecture proposed in the original paper, which is already close in spirit to the transformer-based policy network used in our method.

To compare methods on this task, each baseline is first trained using its native design objective, derived from an EIG-based criterion, thereby learning an acquisition policy independently of the downstream predictor. Once this policy is fixed, a downstream network is trained on trajectories generated by the learned policy to predict the source locations from the resulting design–observation history. All methods are then evaluated through the predictive accuracy of this downstream network,

ensuring that the comparison reflects performance on the final source-localization task rather than on the surrogate design objective alone.

The objective of this experiment is threefold. First, we evaluate our method from the perspective of downstream predictive performance. In particular, we assess whether the learned design policy produces informative trajectories that enable the downstream network to accurately solve the source-location point-estimation problem. Second, we evaluate the quality of the learned design policy using the usual lower and upper bounds on the EIG, estimated respectively by sPCE and sNMC. Third, we compare methods in terms of computational cost, measured by the wall-clock training time on a single GPU. We distinguish between the cost of learning the design policy itself and the total training cost, which, for the baselines, also includes the additional training of the downstream prediction network.

Architectures. We use the same policy-network architecture for ACTION-BED and DAD, detailed in Appendix E.1. At the start of each rollout, before any design–outcome observations are available, the initial history encoding is parameterised as a learnable vector. The only architectural hyperparameters varied across methods are the encoding dimensions, which are reported in Table 11

Table 11: **Policy architecture encoding dimensions**.

Method	$d_{\text{enc},\xi}$	$d_{\text{enc},y}$	$d_{\text{enc},r}$	$d_{\text{enc},h}$
DAD	32	32	32	64
ACTION-BED	32	32	32	64

In this task, weight sharing between the design policy and the downstream network did not yield a clear performance improvement without further optimisation. We therefore use a simple fully connected MLP for the downstream network, which maps the design–outcome history $\{(\xi_i, y_i)\}_{i=1}^{T}$ to a point estimate of the source locations. The architecture is detailed in Table 12.

Table 12: Downstream network architecture.

Layer	**Description**	**Dimension**	**Activation**
Input	h_T	$T \times (\dim_\xi + \dim_y) = 90$	–
Hidden layer 1	Fully connected	512	GELU
Hidden layer 2	Fully connected	256	GELU
Hidden layer 3	Fully connected	128	GELU
Output	$\hat{\theta}$	4	–

Training procedure. During training, all policy networks are optimised from data generated online by rollouts of the corresponding design policy. At each iteration, latent parameters θ are sampled from the prior distribution $p(\theta)$, and the policy is rolled out autoregressively over the experimental horizon. The hyperparameters used for the policy networks are reported in Table 13. ALINE is implemented using its native architecture and trained with the hyperparameters reported in the original paper for the same source-location finding task. In particular, ALINE is trained for 100 000 iterations with a joint design policy and posterior approximation network, after an initial burning phase of 20 000 iterations during which only the posterior approximation network is optimised. DAD and Action-BED are implemented using the hyperparameters reported in the same table. For Action-BED, we additionally use a warm-up phase in which the downstream prediction network is trained in isolation to infer the source locations from randomly generated designs. After this warm-up, the design and downstream networks are optimised jointly using a shared optimizer. This pretraining stage provides a more informative learning signal to the design policy at the beginning of joint optimization, improves the coupling between the acquisition policy and the downstream predictor, and empirically leads to convergence toward better optima.

Evaluation procedure. After training, we evaluate the downstream action networks associated with each learned design policy, either jointly trained with the acquisition policy in Action-BED, or trained independently after the design policy has been learned for other baselines. We evaluate downstream predictors using the two downstream losses described above. Evaluation rollouts are

Table 13: **Hyperparameters for source location finding.** Training and optimization hyperparameters used for the policy networks and the downstream prediction networks. ExpLR denotes an exponential learning-rate scheduler. The RANDOM downstream network uses the same hyperparameters as DAD.

DAD

Field	Value
Policy training	
Batch size	1024
Contrastive samples L	2000
Policy steps	150K
Policy seed	42
Policy optimization	
Optimizer	Adam
Adam betas	$(0.8, 0.998)$
Policy LR	10^{-3}
Scheduler	ExpLR
Decay γ	0.95
Decay period	2000
Weight decay	0
Downstream training	
Downstream steps	150K
Downstream seed	42
Downstream LR	5×10^{-4}
Downstream sched.	ExpLR

ALINE

Field	Value
Policy training	
Batch size	200
Policy steps	100K
Burn-in steps	20K
Burn-in training	Posterior only
Query pool, burn-in	30
Query pool, train	200
Query proposal	Uniform
Reward discount	1
Policy weight α	1
Policy seed	123
Policy optimization	
Optimizer	Adam
Adam betas	$(0.9, 0.999)$
Policy LR	10^{-3}
Policy sched.	Cosine
Grad. clipping	$\|\nabla\|_\infty \leq 1$
Downstream training	
Downstream steps	100K
Downstream LR	5×10^{-4}
Downstream sched.	ExpLR
Downstream decay γ	0.95
Decay period	2000
Downstream seed	42

Action-BED

Field	Value
Joint training	
Batch size	2000
Joint steps	150K
Warm-up steps	50K
Warm-up training	Downstream only
Warm-up designs	Random
Seed	42
Optimization	
Optimizer	Adam
Adam betas	$(0.8, 0.998)$
Policy LR	5×10^{-4}
Scheduler	ExpLR
Decay γ	0.95
Decay period	2000
Weight decay	0

generated by first sampling latent source parameters $\theta \sim p(\theta)$ from the prior and then rolling out the corresponding adaptive design policy autoregressively over the full experimental horizon. The downstream network receives the resulting terminal history h_T and outputs a point prediction of the source configuration. All reported downstream losses are computed over 2048 independent evaluation rollouts, and uncertainty is reported as ± 1 standard error in Table 1.

EIG-bound evaluation. For each trained design policy, we additionally estimate lower and upper bounds on the EIG. This provides a standard, task-agnostic measure of design quality commonly used in the BED literature. It is not the downstream objective considered in our experiment, but DAD and ALINE are explicitly trained with information-theoretic objectives related to EIG maximisation. We recall the definition of these bounds:

$$\mathcal{L}_{\mathrm{sPCE}} = \mathbb{E}\left[\log \frac{p(h_T \mid \theta_0; \pi_d^\phi)}{\frac{1}{L+1}\sum_{\ell=0}^{L} p(h_T \mid \theta_\ell; \pi_d^\phi)}\right], \tag{58}$$

$$\mathcal{U}_{\mathrm{sNMC}} = \mathbb{E}\left[\log \frac{p(h_T \mid \theta_0; \pi_d^\phi)}{\frac{1}{L}\sum_{\ell=1}^{L} p(h_T \mid \tilde{\theta}_\ell; \pi_d^\phi)}\right], \tag{59}$$

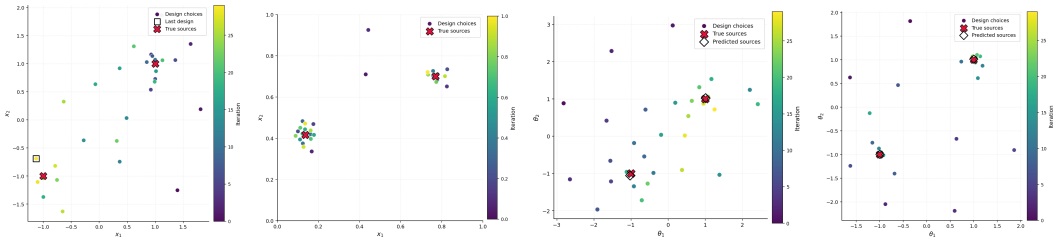
with $\theta_0 \sim p(\theta)$, $\theta_{1:L} \sim p(\theta)$ are independent Monte Carlo samples used to approximate the marginal likelihood, and L being the number of contrastive samples drawn. For all methods, we use a large number of contrastive samples in both estimators so that the resulting bounds are sufficiently tight for comparison, see Table 14. Reported uncertainties correspond to ± 1 standard error over independent rollouts.

G.1.1 Qualitative Results

We complement the quantitative evaluation with qualitative trajectory plots of the learned design policies. We roll out each trained policy and visualize the sequence of selected designs over the experimental horizon. This comparison illustrates the characteristic behavior induced by each method: how quickly it explores the design space, whether it concentrates measurements around informative regions, and how its sampling strategy adapts as observations are collected. These trajectories can provide an interpretable diagnostic of the experimental strategies learned by the different policies.

Table 14: **Source location finding.** Evaluation protocol for design policies. L denotes the number of contrastive samples used to estimate the sequential PCE and NMC bounds. The corresponding batch size is the number of rollout histories used for these EIG-bound estimates. Downstream losses are evaluated on independent rollout batches.

Method	**Training objective**	**sPCE / sNMC**		**Downstream losses**
		L	**Batch size**	**Batch size**
Action-BED (MSE)	Downstream loss	10^6	2000	2000
Action-BED (QLL)	Downstream loss	10^8	200	2000
DAD	EIG-oriented objective	10^6	2000	2000
ALINE	EIG-oriented objective	5×10^6	500	2000
Random	None	10^5	2000	2000



(a) DAD (b) ALINE (c) ACTION-BED (MSE) (d) ACTION-BED (QLL)

Figure 4: **Comparison of four policy rollouts.** Each panel shows one experimental design trajectory under the trained design policy.

G.1.2 Amortising Downstream Bayes-action Selection

This section details the experimental protocol used to compare trained design policies on the downstream task, in terms of the quality of the Bayes-optimal action induced by the experimental trajectories they generate. This serves two purposes. First, it provides a direct way to compare design policies under the decision-theoretic objective of interest, namely the losses incurred by the action selected after observing the experimental history. Second, it illustrates a key practical advantage of ACTION-BED: since the design policy is trained jointly with an amortised downstream action policy, it directly learns an action network that predicts a near-optimal decision, thereby amortising the otherwise costly posterior-inference step. Indeed, for design policies trained on the EIG, the downstream action-selection step is not directly available from training. Therefore, in order to evaluate such policies on downstream performance, one must proceed in two stages: first generate histories using the fixed trained design policy, and then infer or learn the corresponding downstream action from these histories. This is precisely the comparison performed here.

In this experiment, we evaluate the posterior Fréchet mean under both the MSE and QLL losses. For the MSE objective, this estimator coincides with the Bayes action; see Appendix F. This provides a useful explicit posterior-based decision rule against which amortised downstream prediction can be compared. For each trained design policy, we compute the posterior Fréchet mean from posterior samples conditioned on the observed trajectory, and report its empirical downstream performance over the evaluation batch under both loss functions.

Inference protocol. To compute the explicit posterior Bayes action, we consider three posterior-mean estimators. The first one is based on importance sampling with the prior as proposal distribution. For each experimental history $h_T = (\xi_{1:T}, y_{1:T})$ generated by a fixed design policy, we sample particles $\theta^{(1)}, \ldots, \theta^{(N)} \sim p(\theta)$ independently from the prior distribution. These particles are then reweighted according to the likelihood of the observed history,

$$w^{(i)}(h_T) \;\propto\; p(h_T \mid \theta^{(i)}), \qquad \sum_{i=1}^{N} w^{(i)}(h_T) = 1.$$

This yields a weighted empirical approximation of the posterior distribution,

$$p(\theta \mid h_T) \approx \sum_{i=1}^{N} w^{(i)}(h_T)\, \delta_{\theta^{(i)}}.$$

However, because the latent sources are exchangeable, the direct posterior mean is not Bayes-optimal for the permutation-invariant MSE loss considered and may yield a degenerate estimate of the underlying sources by averaging across permutations. To account for this symmetry, we first optimally permute the sources of each particle. Given a reference ordering $\bar{\theta}$, for instance the current empirical estimate or an initial point estimate, each particle $\theta^{(i)} = (\theta_1^{(i)}, \theta_2^{(i)})$ is aligned by solving

$$\sigma_i^\star \in \arg\min_{\sigma \in \mathcal{S}_2} \sum_{k=1}^{2} \left\| \theta_{\sigma(k)}^{(i)} - \bar{\theta}_k \right\|^2,$$

where $\mathcal{S}_K$ denotes the set of permutations of the latent sources. We then define the aligned particle

$$\widetilde{\theta}^{(i)} = \left(\theta_{\sigma_i^\star(1)}^{(i)}, \theta_{\sigma_i^\star(2)}^{(i)} \right),$$

and estimate the MSE Bayes action by the weighted posterior Fréchet mean

$$\widehat{a}_{\mathrm{IS}}(h_T) = \sum_{i=1}^{N} w^{(i)}(h_T)\, \widetilde{\theta}^{(i)}.$$

This aligned posterior mean is then used as the downstream prediction, and its empirical loss is evaluated on the held-out batch. The numerical parameters used for this importance-sampling estimator are reported in Table 15.

The second estimator is based on Hamiltonian Monte Carlo (HMC) [58]. HMC constructs a Markov chain targeting the posterior distribution by simulating Hamiltonian dynamics in an augmented position–momentum space. For each history h_T, we use HMC to draw approximate samples from the posterior distribution. In practice, the HMC chain is initialised from a coarse approximation of the optimal action obtained by a lighter version of the prior-proposal importance-sampling procedure described above. This provides a reasonable starting point near a high-posterior-probability region and stabilises the subsequent sampling stage. After burn-in and thinning according to the parameters reported in Table 15, we obtain posterior samples for each observed trajectory. The HMC-based Bayes action is then computed using the same post-processing procedure as for importance sampling. The empirical posterior mean is then computed over the aligned samples and used as the downstream MSE Bayes-action estimate. We finally evaluate the empirical downstream MSE incurred by this prediction over the evaluation batch.

The third estimator is based on a Laplace approximation of the posterior distribution. For each history h_T, we first compute a maximum-a-posteriori estimate of the latent parameters by optimising the log posterior,

$$\widehat{\theta}_{\mathrm{MAP}}(h_T) \in \arg\max_{\theta} \left\{ \log p(h_T \mid \theta) + \log p(\theta) \right\}.$$

In practice, the optimisation is initialised from a coarse particle-based approximation in order to start near a high-posterior-probability region, and multiple restarts are used to improve robustness. The posterior is then locally approximated by a Gaussian distribution centred at the MAP, with precision given by the negative Hessian of the log posterior at this point,

$$p(\theta \mid h_T) \approx \mathcal{N}\left(\widehat{\theta}_{\mathrm{MAP}}(h_T), \left(H(h_T) + \lambda_{\mathrm{ridge}} I \right)^{-1} \right),$$

where $H(h_T)$ denotes the local posterior precision and λ_{ridge} is a small ridge regularisation parameter added for numerical stability. We then draw samples from this Gaussian approximation using a Cholesky decomposition whenever the regularised precision matrix is positive definite, as described in Table 15. The Laplace-based Bayes action is finally computed using the same post-processing procedure as for importance sampling and HMC: the Gaussian samples are aligned to account for the exchangeability of the latent sources, and the empirical posterior mean over the aligned samples is used as the downstream MSE Bayes-action estimate. We finally evaluate the empirical downstream MSE incurred by this prediction over the evaluation batch.

(a) Hamiltonian Monte Carlo

Parameter	Value
Evaluation batch size	2000
HMC samples	5×10^4
Burn-in steps	10^4
Step size $\epsilon^\star$	$[10^{-3}, 10^{-2}]$
Leapfrog steps	10
Thinning interval	1
PM init. particles N_{PM}	10^5

(b) Laplace approximation

Parameter	Value
Evaluation batch size	2000
MAP optimisation steps	1000
Laplace samples	5×10^4
Ridge regularisation λ_{ridge}	10^{-3}
Factorisation	Cholesky Decomposition

(c) Prior-proposal IS

Parameter	Value
Particles N	5×10^6
Evaluation batch size	2000

Table 15: **Hyperparameters for posterior Bayes-action estimation.** We report the parameters used for the three Bayes-action estimators: prior-proposal IS, Laplace and HMC. The HMC step size ϵ is adapted separately for each method so as to match a target acceptance rate, while remaining constrained to the interval reported in the table.

Indicative inference cost. Since posterior-action estimators require test-time inference for each observed trajectory, we additionally report, in Table 17, the wall-clock time required to infer actions and evaluate the downstream loss on a batch of 2000 trajectories. These timings were obtained on the evaluation machine used for this posterior-inference study, which differs from the hardware used for training. They should therefore be interpreted as indicative deployment costs rather than hardware-normalised runtime benchmarks. Nevertheless, they highlight the qualitative distinction between amortised action prediction, which only requires a forward pass through the downstream network, and posterior-inference baselines, which require iterative or sampling-based computation at test time.

Loss-dependent acquisition behaviour. The results reported in Table 16 highlight an important distinction between the two downstream losses considered in this work. Although both losses evaluate the quality of a final action inferred from the experimental history, they encode different downstream objectives. Consequently, the corresponding Bayes actions are not the same, and neither are the design policies learned by ACTION-BED. This distinction is crucial when interpreting the posterior-action evaluations of fixed design policies. The experimental trajectories produced by the policy are shaped by the final decision criterion they are meant to serve. ACTION-BED trained with the QLL, as shown in Table 1, achieves strong QLL performance, and its learned designs often concentrate tightly around the latent sources. One might therefore expect that, under these highly localised designs, the MSE Bayes action obtained by posterior inference would also be especially accurate, perhaps even outperforming the policies trained with other objectives. However, this is not what we observe in practice.

As illustrated in Figure 5, and quantified in Table 18, the distribution of rollout-wise MSE losses differs substantially between ACTION-BED (MSE) and ACTION-BED (QLL). Under the same evaluation conditions, ACTION-BED (QLL) achieves markedly lower losses for a large fraction of rollouts with a median of 0.008×10^{-2} compared to 0.27×10^{-2} for the MSE-trained policy. However, this improvement comes at the cost of a much heavier upper tail: at the 99th percentile, its MSE reaches 95.9×10^{-2}, compared to only 5.8×10^{-2} under MSE training. Thus, although the QLL-trained policy produces very accurate predictions on the majority of trajectories, it also induces rare rollouts on which the downstream action can fail severely. These high-loss trajectories dominate the average MSE and explain why the overall MSE performance of the QLL-trained policy is worse despite its stronger median behaviour.

This behaviour is consistent with the different training signals induced by the two losses. The logarithmic component of QLL strongly rewards reductions in already small prediction errors, encouraging precise localisation whenever achievable. In contrast, the quadratic objective places

Table 16: **Deployment-time Bayes-action estimation under fixed acquisition policies.** For each pretrained design policy, we compare amortised action networks with posterior-based Bayes-action estimators to assess downstream decision-making from the acquired histories. Retrained networks are trained post-hoc on histories generated by the fixed design policy to minimise the relevant loss, whereas native ACTION-BED networks are jointly trained with their design policies. Uncertainties are reported as one standard error over 2048 rollouts.

Acquisition	**Decision Rule**	**Performance** ($\pm$ 1 s.e.)	
		MSE (10^{-2}) $\downarrow$	QLL $\downarrow$
ACTION-BED (MSE)	Native network (MSE)	0.55 $\pm$ 0.06	-6.83 $\pm$ 0.03
	Retrained network (MSE)	0.86 $\pm$ 0.09	-6.93 $\pm$ 0.01
	Bayes-act. (IS)	0.44 $\pm$ 0.03	-7.25 $\pm$ 0.03
	Bayes-act. (HMC)	0.61 $\pm$ 0.04	-7.11 $\pm$ 0.02
	Bayes-act. (Laplace)	0.46 $\pm$ 0.03	-7.33 $\pm$ 0.04
ACTION-BED (QLL)	Native network (QLL)	2.8 $\pm$ 0.4	-10.10 $\pm$ 0.03
	Retrained network (QLL)	1.7 $\pm$ 0.2	-9.97 $\pm$ 0.03
	Bayes-act. (IS)	1.3 $\pm$ 0.6	-9.37 $\pm$ 0.04
	Bayes-act. (HMC)	2.00 $\pm$ 0.43	-10.12 $\pm$ 0.04
	Bayes-act. (Laplace)	2.31 $\pm$ 0.71	**-10.32 $\pm$ 0.05**
DAD	Retrained network	5.6 $\pm$ 0.6	-6.58 $\pm$ 0.05
	Bayes-act. (IS)	3.9 $\pm$ 0.2	-8.06 $\pm$ 0.05
	Bayes-act. (HMC)	3.6 $\pm$ 0.5	-8.11 $\pm$ 0.05
	Bayes-act. (Laplace)	3.5 $\pm$ 0.8	-8.05 $\pm$ 0.06
ALINE	Retrained network	0.21 $\pm$ 0.03	-8.11 $\pm$ 0.07
	Bayes-act. (IS)	0.16 $\pm$ 0.05	-9.22 $\pm$ 0.03
	Bayes-act. (HMC)	0.35 $\pm$ 0.21	-9.17 $\pm$ 0.03
	Bayes-act. (Laplace)	**0.13 $\pm$ 0.04**	-9.74 $\pm$ 0.07
Random	Retrained network	22.4 $\pm$ 0.8	-3.16 $\pm$ 0.04
	Bayes-act. (IS)	6.1 $\pm$ 0.7	-4.36 $\pm$ 0.05
	Bayes-act. (HMC)	15.7 $\pm$ 2.5	-3.45 $\pm$ 0.08
	Bayes-act. (Laplace)	32.4 $\pm$ 3.1	-3.28 $\pm$ 0.09

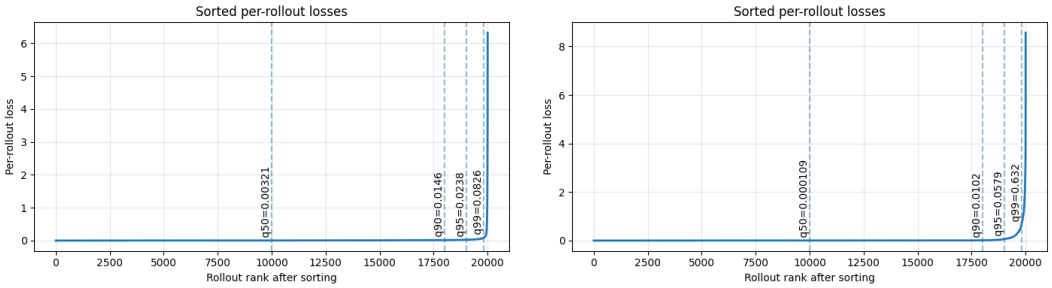
Table 17: **Indicative test-time cost of posterior-action inference.** We report the approximate wall-clock time required to infer actions and evaluate the downstream loss on a batch of 2000 trajectories. Timings are measured on a separate evaluation machine and are intended only to indicate the approximate order of magnitude of the deployment cost.

Action Estimator	**Approx. time (seconds)** $\downarrow$
Amortised downstream network	$\approx$ **6**
Importance-sampling posterior mean	$\approx$ 7600
Laplace posterior approximation	$\approx$ 250
HMC posterior approximation	$\approx$ 3100

greater emphasis on large residuals, encouraging robustness against severe failures. This trade-off also helps interpret the higher EIG bounds achieved by ACTION-BED (QLL) in Figure 2: its objective appears to encourage acquisition strategies that support highly accurate source identification, even though EIG is not directly optimised. Importantly, the two variants optimise different downstream objectives, rather than competing to minimise the same loss. The posterior-based evaluations in Table 16 further show that the QLL-trained acquisition policy remains less effective under the mean MSE criterion, even when using posterior-based MSE Bayes-action estimators instead of its amortised action network.

G.1.3 Additional Results

Training time decomposition. All training-time comparisons are performed on the same GPU, so that reported differences reflect the computational cost of the methods rather than hardware variability. In addition to the main performance metrics, we report in Table 19 a decomposition of the training time into policy and downstream action-network training. This decomposition shows that DAD has a slightly larger total training time than Action-BED, although its policy network alone is faster to train than the jointly trained Action-BED model. This is likely due to the batch sizes used in this experiment and to the relatively large neural architectures, for which backpropagation represents a substantial fraction of the total cost. Consequently, the computational advantage of the singly



(a) ACTION-BED (MSE) (b) ACTION-BED (QLL)

Figure 5: **Distribution of rollout-wise MSE losses for loss-specialised ACTION-BED policies.** We report per-rollout downstream MSE losses sorted from smallest to largest for ACTION-BED trained with the MSE objective and with the QLL objective over 20000 rollouts. The QLL-trained policy achieves smaller errors on most rollouts but exhibits a substantially heavier upper tail, as further quantified in Table 18.

Table 18: **Source Location Finding: Quantiles of per-rollout downstream losses.** For each trained policy, we evaluate 4096 rollouts and compute the loss quantiles separately for each evaluation metric.

Method	Evaluation loss	q_{50}	q_{90}	q_{95}	q_{99}
ACTION-BED (MSE)	MSE (10^{-2})	0.27	1.1	**1.9**	**5.8**
	QLL	-6.83	-5.37	-4.91	-3.96
ACTION-BED (QLL)	MSE (10^{-2})	**0.008**	**0.72**	4.2	95.9
	QLL	**-10.43**	**-7.43**	**-6.23**	**-4.12**

intractable objective is partly diluted by the cost of the network updates. ALINE is markedly more expensive: policy training alone already exceeds the full joint-training time of Action-BED, and its downstream stage remains computationally demanding because adaptive rollouts are costly to generate. Due to its simplicity, the random-design baseline is comparatively inexpensive to train.

Evolution of downstream performance. Figure 6 reports an additional time-resolved comparison of downstream performance during policy training. For each method, we periodically freeze the design-policy weights and evaluate the downstream losses using the same evaluation protocol as above. For DAD and ALINE, this requires training a downstream action network from scratch at each frozen policy checkpoint, using the same architecture and training hyperparameters as reported in Tables 12 and 13. For Action-BED, we instead evaluate the downstream action network associated with the joint model at the corresponding checkpoint. Importantly, the horizontal axis only accounts for the elapsed policy-training time, or joint-training time for Action-BED, including the potential warm-up time. Thus, it excludes the additional cost of training independent downstream action networks for the baselines. Checkpoints are shown up to convergence. For the joint Action-BED models, both downstream losses are evaluated at every checkpoint, even when the downstream predictor was not explicitly optimised for the loss under consideration.

This experiment illustrates how quickly each design policy produces histories that are useful for downstream prediction. Under the permutation-invariant MSE, Action-BED rapidly reaches a regime in which its action network performs well, achieving the best performance during approximately the first six hours of training. At later checkpoints, however, downstream networks trained on ALINE policies achieve stronger performance. Under the QLL, Action-BED trained with the QLL objective yields substantial gains very early in training, outperforming the other methods by a large margin.

Effect of Joint Training on Policy Co-Adaptation. We further examine whether joint training induces a useful co-adaptation between the design policy network and the downstream action policy network. To isolate this effect, we take a fully trained ACTION-BED design policy, keep it fixed, and retrain a new downstream action network under the same optimisation budget and hyperparameters as in the original joint training procedure; see Tables 12 and 13. We perform this retraining separately with the MSE and QLL objectives.

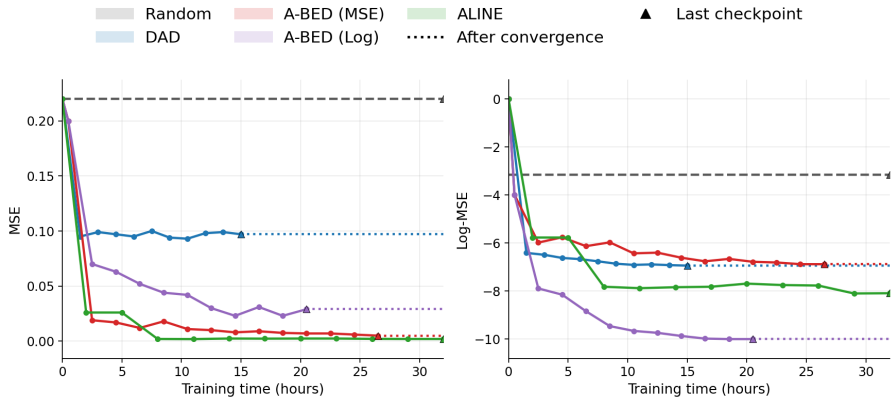


Figure 6: **Comparison of downstream losses** as a function of design policy training time, until convergence.

Table 19: **Training Time Decomposition**. Wall-clock GPU times are reported in minutes. For ACTION-BED, design and downstream action networks are trained jointly, so only total training time is reported.

Method	Design	Downstream	Total
ACTION-BED (MSE)	–	–	1171
ACTION-BED (QLL)	–	–	1184
DAD (MSE)	879	335	1214
DAD (QLL)	879	337	1216
ALINE (MSE)	1780	1182	2962
ALINE (QLL)	1780	1190	2970
Random	0	31	31

The relevant comparison is the same-loss setting: for each ACTION-BED model, we compare the native jointly trained action network to a post-hoc action network retrained with the same loss used to learn the design policy. As shown in Table 20, this retrained network performs slightly worse than the native joint network, despite being trained on histories generated by an already fully optimised design policy. This suggests that the gain of joint training is due to the optimisation path through which the design and action networks become mutually adapted. In particular, the design policy learns to generate histories that are informative in a form the action network can exploit, while the action network simultaneously learns to decode both the observed outcomes and the information implicit in the adaptive sequence of designs.

Table 20: **Native versus retrained downstream networks for Action-BED.** Native networks are obtained directly from joint Action-BED training. Retrained networks are trained post-hoc after freezing the learned acquisition policy, using the same downstream loss. Uncertainties are reported as one standard error.

Acquisition	Action Network	Performance ($\pm$ 1 s.e.)	
		MSE (10^{-2}) $\downarrow$	QLL $\downarrow$
ACTION-BED (MSE)	Native network	$\mathbf{0.55 \pm 0.06}$	-6.83 $\pm$ 0.03
	Retrained network	0.86 ± 0.09	-6.93 $\pm$ 0.01
ACTION-BED (QLL)	Native network	2.8 ± 0.4	$\mathbf{-10.10 \pm 0.03}$
	Retrained network	1.7 ± 0.2	-9.97 $\pm$ 0.03

Cross-Evaluation of Design and Action Policies. We further study the coupling between acquisition and decision-making by cross-evaluating all trained downstream action networks across all design policies in the benchmark. For each trained design policy, we generate histories under that policy and evaluate the predictions produced by action networks trained with every other method with the MSE loss. For ACTION-BED, the action network is the one learned jointly with the design policy;

for the other baselines, it is the network trained subsequently on trajectories from the fixed design policy. Table 21 reports normalised performance ratios, measuring, for each fixed design policy, the degradation incurred when replacing its associated action network by an action network trained on another history distribution.

This experiment tests whether action policies learn transferable Bayes-optimal action, or instead become specialised to the distribution of histories induced by their own design policy. Design policies that are more accurate and concentrate their measurements tightly around the latent sources induce history distributions on which action networks trained under other acquisition strategies perform substantially worse. In this regime, the downstream predictor must exploit fine-grained, policy-specific patterns in the adaptive design sequence, rather than only aggregate generic information from the observations. Conversely, the action networks paired with these more focused design policies achieve strong performance on their own histories but transfer poorly to histories generated by other policies. This suggests that high-performing methods do not merely collect more information; they also shape the information in a form that is specifically matched to their associated downstream decision rule.

Table 21: **Normalized cross-policy coupling between acquisition and action networks for the MSE loss.** Each entry reports the normalized degradation factor for a test acquisition policy π_d^i paired with an action network π_a^j, defined as $\widetilde{R}_{ij} = R(\pi_d^i, \pi_a^j)/\min_k R(\pi_d^i, \pi_a^k)$. Rows indicate the acquisition policies used at test time, and columns the downstream action networks.

Test Acquisition	**Downstream Action Network**				
	A-BED (MSE)	A-BED (QLL)	DAD	ALINE	Random
ACTION-BED (MSE)	**1.00**	304.55	114.73	175.64	13.64
ACTION-BED (QLL)	17.11	**1.00**	16.07	34.75	9.21
DAD	19.48	1079.98	**1.00**	16.89	6.14
ALINE	32.86	18.10	20.03	**1.00**	131.91
Random	11.83	1135.97	4.16	10.82	**1.00**

Impact of Warmstarting in Joint Training. We assess whether warmstarting the downstream action network improves joint ACTION-BED training. Table 22 shows that, for both the MSE and QLL objectives, warmstarting leads to better final performance. This suggests that the benefit is not limited to a faster initial optimisation of the action network. Rather, by providing a more accurate downstream predictor early in training, warmstarting yields a more informative learning signal for the design policy. This improved signal changes the subsequent joint optimisation dynamics and guides the coupled design–action system toward a better final solution.

Table 22: **Impact of warmstarting in joint Action-BED training.** Uncertainties are reported as one standard error.

Acquisition	**Warmstart**	**Performance** ($\pm$ 1 s.e.)				**Compute Time**
		MSE (10^{-2}) $\downarrow$	QLL $\downarrow$	sPCE $\uparrow$	sNMC $\uparrow$	GPU (min) $\downarrow$
ACTION-BED (MSE)	Yes	$\mathbf{0.55 \pm 0.06}$	-6.83 ± 0.03	11.63 ± 0.04	12.84 ± 0.08	1171
	No	0.87 ± 0.1	-6.82 ± 0.01	11.62 ± 0.08	12.67 ± 0.15	**1159**
ACTION-BED (QLL)	Yes	2.8 ± 0.4	$\mathbf{-10.10 \pm 0.03}$	$\mathbf{17.14 \pm 0.11}$	$\mathbf{21.07 \pm 0.45}$	1184
	No	3.0 ± 0.3	-9.33 ± 0.03	16.48 ± 0.18	18.88 ± 0.49	1172

Training-seed robustness of the main architectures. To further assess the robustness of our results to training initialization, we repeat the training of ACTION-BED and DAD over five independent random seeds using the main architectures on two benchmarks: source location finding and the stochastic single pendulum. The results are reported in Tables 23 and 24, respectively, as mean $\pm$ standard error across training seeds, rather than across evaluation rollouts as in the main results.

Across both tasks, the results exhibit relatively limited variability across training seeds and preserve the main downstream-performance trends. In particular, each ACTION-BED variant achieves the lowest average loss under its corresponding training objective, while the QLL-trained policies consis-

Table 23: **Source Location Finding: Training-seed variability.** Mean ± standard error over five independent training seeds using the main architectures. ACTION-BED is evaluated with its native action network on the relevant loss, whereas for DAD an action network is retrained post-hoc on the relevant loss and evaluated under the same loss.

Method	MSE (10^{-2}) ↓	QLL ↓	sPCE ↑	sNMC ↑
ACTION-BED (MSE)	$\mathbf{0.63 \pm 0.05}$	-6.91 ± 0.07	11.72 ± 0.12	12.73 ± 0.22
ACTION-BED (QLL)	3.57 ± 0.12	$\mathbf{-10.11 \pm 0.08}$	$\mathbf{17.49 \pm 0.23}$	$\mathbf{20.13 \pm 0.62}$
DAD	5.12 ± 0.48	-6.93 ± 0.09	13.01 ± 0.09	17.13 ± 0.14

Table 24: **Stochastic Single Pendulum: Training-seed variability.** Mean ± standard error over five independent training seeds using the main architectures, and with the same protocol as in Table 23.

Method	MSE (10^{-3}) ↓	W. MSE (10^{-3}) ↓	QLL ↓	sPCE ↑	sNMC ↑
ACTION-BED (MSE)	$\mathbf{6.2 \pm 0.2}$	4.8 ± 0.1	-6.57 ± 0.01	3.66 ± 0.01	3.69 ± 0.03
ACTION-BED (QLL)	7.2 ± 0.1	5.3 ± 0.4	$\mathbf{-6.73 \pm 0.02}$	$\mathbf{3.96 \pm 0.02}$	$\mathbf{3.97 \pm 0.04}$
ACTION-BED (W. MSE)	13.3 ± 1.7	$\mathbf{3.6 \pm 0.4}$	-5.85 ± 0.09	3.16 ± 0.07	3.19 ± 0.05
DAD	7.7 ± 0.4	5.3 ± 0.5	-6.37 ± 0.02	3.89 ± 0.02	3.90 ± 0.02

tently attain strong EIG bounds. These findings provide additional evidence that the performance gains observed in the main experiments are not specific to a single training initialization.

Due to the substantial computational cost of training the full architectures, this analysis is restricted to ACTION-BED and DAD on two representative benchmarks. In particular, we do not repeat the training of ALINE or the remaining tasks across multiple seeds. Nevertheless, these additional experiments complement the training-seed stability analysis conducted with smaller architectures in Appendix G.4, and provide further evidence of the robustness of ACTION-BED.

G.2 Dynamical Systems

We consider sequential design problems in which the underlying simulator is a dynamical system. Unlike exchangeable observation models, each outcome depends on the current system state, so the order in which observations are collected matters. This setting is adapted from [41] and covers both the simple- and double-pendulum experiments considered in this work.

More precisely, we view these experiments as an abstraction of dynamical systems governed by stochastic differential equations, detailed below, whose observation model is Markovian:

$$p(y_{t+1} \mid y_t, \xi_{t+1}, \theta),$$

so that the next observation is generated from the previous state or observation y_t, the current design ξ_{t+1}, and the physical parameters θ. Hence, the likelihood is sequential and cannot be factorized as with conditionally exchangeable observations. Under these assumptions, the joint distribution of a trajectory can be written as

$$\begin{aligned} p(h_T \mid \theta; \pi_d^\phi) &= \prod_{t=1}^{T} p(y_t \mid h_{t-1}, \theta) \\ &= \prod_{t=1}^{T} p(y_t \mid y_{t-1}, \xi_t, \theta). \end{aligned}$$

Architectures. For both pendulum tasks, we use the same TNP-like policy architecture for ACTION-BED and DAD, as described in Appendix E.1 and with the same encoding dimensions as in Table 11. Since the likelihood is non-exchangeable, we add a time embedding to each history element r_t before aggregation. At the beginning of a rollout, when no design–outcome pair has yet been observed, the initial history representation is parameterized as a learnable vector.

For both ACTION-BED and DAD, the emitter output is additionally mapped through an affine–$\tanh$ transformation,

$$\xi_t = a \tanh(\tilde{\xi}_t) + b,$$

which ensures that the selected designs lie in the admissible control domain. The same architectural template is used for the stochastic and double pendulum experiments; only the input and output dimensions can vary, depending on the dimension of the designs and observations.

ALINE is permutation-invariant by construction and is therefore not directly tailored to non-exchangeable likelihoods. To make the sequential structure explicit, we augment each context element with a time encoding and include the most recent observation in the context before encoding. This provides the model with the information needed to condition each candidate design on the current dynamical state. RANDOM samples designs uniformly at random over the valid constrained space.

In all cases, the downstream predictor is a fully connected MLP mapping the complete design–outcome history to a point estimate of θ in the relevant space (see Table 25). No parameter sharing is used between the design policy and the downstream network.

Table 25: Downstream network architecture.

Layer	**Description**	**Dimension**	**Activation**
Input	h_T	$T \times (\dim_\xi + \dim_y)$	–
Hidden layer 1	Fully connected	512	GELU
Hidden layer 2	Fully connected	256	GELU
Hidden layer 3	Fully connected	128	GELU
Output	$\hat{\theta}$	d_θ	–

Training procedure. For the dynamical-system experiments, we follow a training protocol closely mirroring that of the source location finding task. All policies are trained from trajectories generated by the corresponding simulator. At each iteration, latent parameters θ are sampled from the prior, the design policy is rolled out autoregressively over the experimental horizon, and each method is trained on its own objective. We compare several downstream objectives in order to assess the accuracy and flexibility of the learned predictive action network. We consider the standard mean-squared error on θ,

$$\ell_{\mathrm{MSE}}(\hat{\theta}, \theta) = \frac{1}{d_\theta} \|\hat{\theta} - \theta\|_2^2,$$

the logarithmic variant,

$$\ell_{QLL}(\hat{\theta}, \theta) = \frac{1}{d_\theta} \sum_{j=1}^{d_\theta} \left[\log \left[(\hat{\theta}_j - \theta_j)^2 + \epsilon \right] + (\hat{\theta}_j - \theta_j)^2 \right],$$

and a weighted MSE,

$$\ell_{\mathrm{wMSE}}(\hat{\theta}, \theta) = \frac{1}{d_\theta} \sum_{j=1}^{d_\theta} w_j (\hat{\theta}_j - \theta_j)^2.$$

For the stochastic pendulum, we use weights

$$w = (0.1, 1.0, 2.0),$$

whereas for the double pendulum we use

$$w = (1.0, 1.0, 2.0, 3.0).$$

The weighted objective allows different components of θ to be emphasized, illustrating that the downstream target can be adapted to the scientific or operational quantity of interest.

Evaluation procedure. In addition to downstream prediction performance, we evaluate each trained policy using sequential PCE and sequential NMC bounds on the EIG, which provide a common information-theoretic measure of design quality across methods. For both dynamical systems and for all downstream losses, prediction metrics are estimated from 2048 rollout batches and reported with ± 1 standard error. For the single-pendulum experiment, the sequential PCE and NMC estimates are computed with $L = 50$K contrastive samples and a batch size of 2000. For the double-pendulum experiment, we use a larger contrastive budget, with $L = 200$K and a batch size of 1000. We also report the total GPU compute time required for training.

G.2.1 Stochastic Pendulum

The first dynamical system considered is a stochastic single pendulum. The simulator describes the trajectory of a pendulum whose state evolves over time according to noisy dynamics. At each step, the design variable corresponds to an external control applied to the system, allowing the policy to influence the subsequent motion of the pendulum and thereby collect informative observations about the latent physical parameters.

Task. We consider the problem of actively identifying the physical parameters of a stochastic controlled pendulum. The state is $x_t = (q_t, \dot{q}_t)$, where q_t denotes the angle from the vertical and $\dot{q}_t$ the angular velocity. The parameters of interest are the mass and length of the pendulum, (m, l), while the gravitational acceleration and damping coefficient are fixed to $g = 9.81,\ d = 0.1$. To obtain dynamics that depend linearly on the unknown parameters, we introduce the reparametrization $\theta := \left(\frac{3g}{2l}, \frac{3d}{ml^2}, \frac{3}{ml^2}\right)^\top$. The initial state is fixed as $x_0 = (0, 0)^\top$. At each step of the discretized dynamics, with time step $\Delta t = 0.05$, the policy selects a bounded control input $\xi_t \in [-1, 1]$, corresponding to a torque applied to the pendulum. The observation is the state of the system at the next discretization step. The objective is to choose controls that generate trajectories informative about the latent physical parameters.

Stochastic Dynamics. In continuous time, the controlled dynamics follow the stochastic differential equation

$$dx_t = (\dot{q}_t, h(x_t, \xi_t)^\top \theta)^\top \, dt + L \, d\beta_t,$$

where β_t is a Brownian motion,

$$h(x_t, \xi_t) = (-\sin q_t,\ -\dot{q}_t,\ \xi_t), \qquad L = (0, 0.1)^\top.$$

Equivalently,

$$\begin{aligned} dq_t &= \dot{q}_t \, dt, \\ d\dot{q}_t &= h(x_t, \xi_t)^\top \theta \, dt + 0.1 \, d\beta_t. \end{aligned}$$

The observations used in the experiment are the discretized states

$$y_t = x_t = (q_t, \dot{q}_t),$$

recorded every $\Delta t = 0.05$. The resulting observation process is sequential: each new observation depends on the previous state, the applied control, and the latent parameter θ.

In our setting, we place a Gaussian prior directly on the effective coefficients of θ, without enforcing the constraints associated with a fixed set of physical parameters:

$$p(\theta) = \mathcal{N}\left(\begin{bmatrix} 14.7 \\ 0 \\ 3.0 \end{bmatrix}, \begin{bmatrix} 0.1 & 0 & 0 \\ 0 & 0.01 & 0 \\ 0 & 0 & 0.1 \end{bmatrix} \right).$$

The downstream task is therefore to estimate the latent dynamical coefficients θ from the controlled trajectory. The design variable ξ_t represents the external torque applied to the pendulum.

Hyperparameters. The main training hyperparameters are reported in Table 26. ACTION-BED and DAD are trained for 50K gradient steps. For ACTION-BED, we additionally use a 10K-step warm-up phase in which the downstream network is trained on trajectories generated from random designs. ALINE is trained with a total budget of 100K steps, including a 20K-step burning phase during which only the posterior inference network is optimised.

G.2.2 Stochastic Double Pendulum

We next consider a controlled stochastic double-pendulum system. Compared with the single-pendulum experiment, this benchmark involves higher-dimensional, nonlinear, and coupled dynamics, making the generated trajectories substantially more sensitive to both the applied controls and the latent physical parameters. The state is

$$x_t = \begin{bmatrix} q_t \\ \dot{q}_t \end{bmatrix} = (q_{1,t}, q_{2,t}, \dot{q}_{1,t}, \dot{q}_{2,t})^\top \in \mathbb{R}^4,$$

Table 26: **Hyperparameters for stochastic single pendulum.** Training and optimization hyperparameters used for the policy networks and the downstream prediction networks. ExpLR denotes an exponential learning-rate scheduler. The RANDOM downstream network uses the same hyperparameters as DAD.

DAD

Field	Value
Policy training	
Batch size	512
Contrastive samples L	2000
Policy steps	50K
Policy seed	42
Policy optimization	
Optimizer	Adam
Adam betas	$(0.8, 0.998)$
Policy LR	10^{-4}
Scheduler	ExpLR
Decay γ	0.96
Decay period	400
Weight decay	0
Downstream training	
Downstream steps	50K
Downstream seed	42
Downstream LR	10^{-4}
Downstream sched.	ExpLR

ALINE

Field	Value
Policy training	
Batch size	200
Policy steps	80K
Burn-in steps	20K
Burn-in training	Posterior only
Query pool, burn-in	50
Query pool, train	200
Query proposal	Uniform
Reward discount	1
Policy weight α	1
Policy seed	123
Policy optimization	
Optimizer	Adam
Adam betas	$(0.9, 0.999)$
Policy LR	10^{-3}
Policy sched.	Cosine
Grad. clipping	$\|\nabla\|_\infty \leq 1$
Downstream training	
Downstream steps	50K
Downstream LR	10^{-3}
Downstream sched.	ExpLR
Downstream decay γ	0.95
Decay period	1000
Downstream seed	42

Action-BED

Field	Value
Joint training	
Batch size	512
Joint steps	50K
Warm-up steps	10K
Warm-up training	Downstream only
Warm-up designs	Random
Seed	42
Optimization	
Optimizer	Adam
Adam betas	$(0.8, 0.998)$
Policy LR	10^{-4}
Scheduler	ExpLR
Decay γ	0.96
Decay period	400
Weight decay	0

where q_1, q_2 are the joint angles and $\dot{q}_1, \dot{q}_2$ their angular velocities. The unknown parameter is

$$\theta = (m_1, m_2, l_1, l_2)^\top,$$

corresponding to the masses and lengths of the two links. As in the single-pendulum setting, observations are the $T = 50$ discretized states of the system, collected along a trajectory. The design variable is a pair of bounded torques

$$\xi_t = (\xi_{1,t}, \xi_{2,t})^\top, \qquad \xi_{1,t} \in [-4, 4], \qquad \xi_{2,t} \in [-2, 2],$$

applied to the two joints. The downstream task is to infer θ from the resulting controlled trajectory.

Stochastic Dynamics. The double-pendulum dynamics follow the standard manipulator form

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} = \xi + \tau_g(q),$$

where $M(q)$ is the inertia matrix, $C(q, \dot{q})$ contains the Coriolis and centrifugal terms, and $\tau_g(q)$ is the gravitational torque. We use $g = 9.81$ and add process noise to the acceleration dynamics. The resulting stochastic differential equation is

$$dq_t = \dot{q}_t\, dt, \qquad d\dot{q}_t = M(q_t)^{-1}\big(\tau_g(q_t) + \xi_t - C(q_t, \dot{q}_t)\dot{q}_t\big)\, dt + L\, d\beta_t,$$

where $\beta_t = (\beta_{1,t}, \beta_{2,t})^\top$ is a two-dimensional Brownian motion with independent components and

$$L = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}.$$

The SDE is discretized with the same Euler–Maruyama scheme as in the single-pendulum experiment, with $\Delta t = 0.05$. The observation process is therefore Markovian and non-exchangeable: each new state depends on the previous state, the applied joint torques, and the latent physical parameters.

We place a log-normal prior on the physical parameters,

$$p(\theta) = \mathrm{LogNormal}\left(\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, 0.01\, I_4\right),$$

so that masses and lengths remain positive. This task therefore tests whether adaptive design policies can select torque sequences that generate trajectories informative about coupled inertial, gravitational, and velocity-dependent effects.

Table 27: **Hyperparameters for stochastic double pendulum.** ExpLR denotes an exponential learning-rate scheduler. The RANDOM downstream network uses the same hyperparameters as DAD.

DAD

Field	Value
Policy training	
Batch size	512
Contrastive samples L	2000
Policy steps	50K
Policy seed	42
Policy optimization	
Optimizer	Adam
Adam betas	$(0.8, 0.998)$
Policy LR	10^{-4}
Scheduler	ExpLR
Decay γ	0.96
Decay period	400
Weight decay	0
Downstream training	
Downstream steps	50K
Downstream seed	42
Downstream LR	10^{-4}
Downstream sched.	ExpLR

ALINE

Field	Value
Policy training	
Batch size	200
Policy steps	80K
Burn-in steps	20K
Burn-in training	Posterior only
Query pool, burn-in	50
Query pool, train	150
Query proposal	Uniform
Reward discount	1
Policy weight α	1
Policy seed	123
Policy optimization	
Optimizer	Adam
Adam betas	$(0.9, 0.999)$
Policy LR	10^{-3}
Policy sched.	Cosine
Grad. clipping	$\|\nabla\|_\infty \leq 1$
Downstream training	
Downstream steps	100K
Downstream LR	10^{-3}
Downstream sched.	ExpLR
Downstream decay γ	0.95
Decay period	1000
Downstream seed	42

Action-BED

Field	Value
Joint training	
Batch size	512
Joint steps	100K
Warm-up steps	20K
Warm-up training	Downstream only
Warm-up designs	Random
Seed	42
Optimization	
Optimizer	Adam
Adam betas	$(0.8, 0.998)$
Policy LR	5×10^{-4}
Scheduler	ExpLR
Decay γ	0.96
Decay period	400
Weight decay	0

Table 28: **Single Pendulum.** Training time decomposition.

Method	Design	Downstream	Total
ACTION-BED (MSE)	–	–	240
ACTION-BED (QLL)	–	–	287
ACTION-BED (W-MSE)	–	–	290
DAD (MSE)	463	207	670
DAD (QLL)	463	246	709
DAD (W-MSE)	463	208	671
ALINE (MSE)	1520	600	2120
ALINE (QLL)	1520	621	2141
ALINE (W-MSE)	1520	602	2122
RANDOM	0	19	19

Hyperparameters. The main training hyperparameters are reported in Table 27. For this task, ACTION-BED is trained for 100K gradient steps, preceded by a 20K-step warm-up phase in which only the downstream action network is optimised. ALINE is trained under the same total budget of 100K steps, including a 20K-step burn-in phase for the posterior inference network. Finally, DAD and its associated downstream networks are trained for 50K steps, as no further improvement was observed beyond this point. In all cases, downstream action networks are trained until their predictive performance saturates.

G.2.3 Additional Results

We report additional compute-time results for both dynamical-system tasks. Tables 28 and 29 decompose the total wall-clock GPU time into design-policy training and downstream-network training whenever these stages are trained separately.

G.3 Masked MNIST Classification Experiment

We consider a sequential masked-classification task inspired by Iollo et al. [39] and based on the MNIST dataset [50]. The latent variable is an image $\theta \in \mathbb{R}^{28\times 28}$, and the target of interest is its class

Table 29: **Double Pendulum: Training time decomposition.** Wall-clock GPU times are reported in minutes. For ACTION-BED, the design and downstream action networks are trained jointly, so only total training time is reported.

Method	Design	Downstream	Total
ACTION-BED (MSE)	–	–	501
ACTION-BED (QLL)	–	–	504
ACTION-BED (W-MSE)	–	–	504
DAD (MSE)	862	72	934
DAD (QLL)	862	78	940
DAD (W-MSE)	862	71	933
ALINE (MSE)	1938	1075	3013
ALINE (QLL)	1938	1124	3062
ALINE (W-MSE)	1938	1069	3007
RANDOM	0	58	58

label $z \in \{0, \dots, 9\}$. The goal is to identify the digit class from a small number of partial and noisy observations of the image.

At each step, the design $\xi \in [1, 28]^2$ specifies a continuous spatial location in pixel coordinates, interpreted as the top-left corner of a local patch. The observation is a noisy 5×5 patch extracted around this location:

$$y = A_\xi \theta + \eta, \qquad \eta \sim \mathcal{N}(0, \sigma^2 I_d),$$

where $d = 5^2$, $A_\xi \theta$ denotes the extracted patch, and zero padding is used whenever the patch extends outside the image domain. This illustrates that the observations are reparametrizable, which allows gradients to be propagated through the observation process. Conditionally on the image, the observation model is therefore explicit:

$$p(y \mid \theta, \xi) = \mathcal{N}(A_\xi \theta, \sigma^2 I_d).$$

A direct implementation of A_ξ as a discrete masking operator would be non-differentiable with respect to ξ: small changes in the queried location would not affect the selected pixels until crossing an integer pixel boundary. To obtain a differentiable simulator, we instead use a continuous counterpart of the patch-extraction operator. In practice, the image is queried by grid sampling at the 5×5 patch coordinates induced by continuous ξ. When a sampling coordinate is not integer-valued, its intensity is obtained by bilinear interpolation from the four neighboring pixels. Thus, moving the design location smoothly changes the observed patch, making the simulator differentiable with respect to ξ while retaining the interpretation of localized pixel measurements.

Over a horizon $T = 5$, the policy adaptively selects patch locations $\xi_{1:T}$ based on the previously observed patches, with the objective of producing a final history informative for classification. This means that the task has a hierarchical structure. Observations are generated from the latent image θ, but the target of interest is the discrete label z. Consequently, the relevant class-conditional observation model is obtained by marginalizing over images within each class,

$$p(y \mid z, \xi) = \int_\Theta p(y \mid \theta, \xi)\, p(\theta \mid z)\, d\theta.$$

Although $p(y \mid \theta, \xi)$ is available in closed form, this class-conditional likelihood is not. In practice, the joint prior over images and labels is only available through the empirical dataset,

$$p(\theta, z) = \frac{1}{N} \sum_{i=1}^{N} \delta_{(\theta_i, z_i)}(\theta, z).$$

The experiment therefore provides a benchmark for sequential design in a high-dimensional, hierarchical, and partially implicit setting. The design problem is challenging because only a small number of localized measurements are allowed, while the latent image is high-dimensional. Informative policies must therefore learn where to observe the image so as to reveal discriminative regions for the downstream classifier. This directly tests whether an acquisition strategy can align its designs with the final prediction task, rather than merely collecting information about the full latent image.

Table 31: **MNIST masked classification hyperparameters: DAD and iDAD.** For iDAD, no additional downstream classifier is trained: predictions are obtained directly from the critic logits. The contrastive denominator is computed exactly over the discrete label space. The RANDOM downstream network uses the same hyperparameters as DAD.

Field	DAD	iDAD
Batch size	512	512
Learning rate	5×10^{-4}	10^{-3}
Policy steps	50K	10K
Contrastive samples	$L = 500$	Exact over labels
Downstream steps	50K	–
Downstream LR	10^{-4}	–

Table 32: **MNIST masked classification hyperparameters: ACTION-BED and ALINE.** For ALINE, predictions are obtained directly from the classification head added to the posterior inference network.

Field	ALINE	Action-BED
Batch size	200	512
Number of queries	128	–
Number of queries, burn-in	128	–
Learning rate	10^{-3}	5×10^{-4}
Total steps	90K	10K
Burn-in steps	10K	–
Warm-up steps	–	0
Joint steps	–	10K

G.3.1 Baselines, architectures, and downstream objective.

Protocol. We follow the same experimental protocol as in the previous experiments, with the main difference that the label-level observation model is implicit. As a result, EIG bounds are not reported for this task. Acquisition policies are trained using only images from the training set, while final performance is evaluated and reported on both the training and test sets. All downstream classifiers share the architecture reported in Table 30 and are trained with the cross-entropy loss

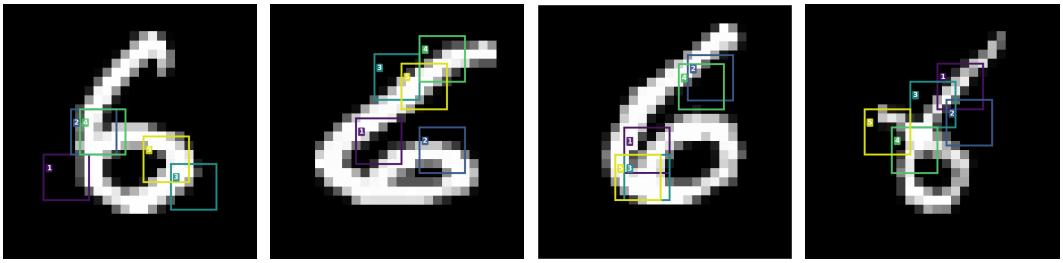
$$\mathcal{L}_{\mathrm{CE}}(\phi, \psi) = -\mathbb{E}_{(z,\theta),h_T}\left[\log \pi_a^{\psi}(z \mid h_T)\right],$$

where z denotes the class label and h_T the final acquisition history. For each method, training is monitored on a validation split and stopped by early stopping once validation performance no longer improves. The resulting training budgets and selected hyperparameters are summarised in Tables 31 and 32. To avoid redundancy, these tables report only the hyperparameters that differ from those in Table 13; all omitted entries are kept unchanged. We report the final classification loss and accuracy on both the training and test sets.

Table 30: **MNIST masked classification downstream architecture**. The downstream classifier uses the same encoder–aggregation structure as the acquisition network, with a larger pair encoder to handle the higher-dimensional masked-image observations. The output dimension $K = 10$ is the number of classes.

Module	Layer	Description	Dimension / Activation
Pair encoder	Input	(ξ_t, y_t)	$\dim(\xi) + \dim(y)$ / –
	H1	Fully connected	512 / ReLU
	H2	Fully connected	256 / ReLU
	H3	Fully connected	128 / ReLU
	Output	Fully connected	32 / –
History aggregation	Input	$\{R(\xi_s, y_s)\}_{s=1}^{T}$	$T \times 32$ / –
	Pooling	Sum or mean pooling	32 / –
Prediction head	Input	Aggregated history	32 / –
	H1	Fully connected	128 / ReLU
	H2	Fully connected	64 / ReLU
	Output	Class logits	K / –

Architectures. The acquisition policies used by DAD, iDAD, and ACTION-BED rely on the same architectural backbone as in Appendix E. Apart from the task-specific dimensions of the design–observation pairs, the encoding dimensions are kept unchanged with respect to Table 11. The only



(a) DAD (b) iDAD (c) ALINE (d) ACTION-BED

Figure 7: **Comparison of four policy rollouts.** Each panel shows one experimental design trajectory under the trained design policy.

additional modification is a sigmoid activation at the output of the emitter network, so that the proposed designs lie in the normalized domain $[0, 1]^2$ before being reprojected onto the pixel grid.

ACTION-BED jointly trains the acquisition policy and downstream classifier for 10K gradient steps using the downstream cross-entropy objective, with no warm-up phase. DAD is trained on sPCE objective, since it requires an explicit likelihood. This objective is defined with respect to the latent image θ rather than the class label z, therefore, its policy seeks designs that discriminate between images, rather than directly between classes. For both DAD and RANDOM, a downstream classifier with the same architecture as in ACTION-BED is then trained separately for 50K gradient steps on trajectories generated by the corresponding policy.

We also compare against iDAD, which adapts the contrastive InfoNCE objective to the implicit class-conditional setting. It jointly learns a design policy π_d^{iDAD} and a critic $U_\psi(z, h_T)$ measuring the compatibility between a candidate label and the observed history. Since the latent parameter space is discrete, the nested expectation appearing in the denominator of the InfoNCE objective can be accurately approximated by summing over all possible labels rather than relying on Monte Carlo negative samples. In practice, this amounts to replacing the usual contrastive denominator by a categorical normalization over the label space,

$$\mathcal{L}_{\text{iDAD}}(\phi, \psi) = -\mathbb{E}_{z,h_T}\left[\log \frac{\exp\{U_\psi(z, h_T)\}}{\sum_{z'=0}^{9} \exp\{U_\psi(z', h_T)\}}\right].$$

For ALINE, we keep the native encoder–decoder architecture and modify only the target prediction head. Whereas the original implementation parameterises the posterior by a Gaussian mixture, the masked classification task requires a discrete posterior over labels. We therefore replace the Gaussian-mixture head by a classification head producing categorical logits. The target decoder outputs embeddings z_{target}, with one target token per class. Each token embedding is passed through a small MLP to produce one scalar logit,

$$\ell_k = f_{\text{cls}}(z_{\text{target},k}), \qquad k = 1, \ldots, 10,$$

and hence a categorical posterior through a softmax. The posterior inference network is trained with the standard cross-entropy loss. This preserves ALINE's native token-based architecture while adapting its output layer to the discrete label space.

Similarly, iDAD already includes a critic network which, in this setting, directly scores class labels as a function of the acquisition history. Its outputs can therefore be interpreted as class logits. Consequently, for both iDAD and ALINE, we do not train an additional downstream classifier: predictions are obtained directly from the critic logits for iDAD and from the classification head logits for ALINE.

Evaluation. For MNIST masked classification, loss and accuracy are evaluated after training on the full training and test sets. Standard errors are computed across examples: for the cross-entropy loss, we report the empirical standard error of the per-example losses; for accuracy, we use the Bernoulli standard error associated with the empirical success rate. These uncertainty estimates therefore reflect finite-sample evaluation variability. Additional Figure 7 shows representative design trajectories for the different methods.

Table 33: **Pair Encoder and Emitter Architectures**.

Module	Layer	Description	Dimension / Activation
Pair encoder	Input	ξ, y	3 / –
	H1	Fully connected	256 / ReLU
	Output	Fully connected	16 / –
Emitter	Input	$R(h_t)$	16 / –
	H1	Fully connected	2 / –
	Output	ξ	2 / –

Table 34: **Source Location Finding hyperparameters**.

Hyperparameter	DAD	Action-BED
Batch size	2000	2000
Contrastive samples L	2000	–
Policy steps	50K	150K
Downstream steps	150K	–
Warm-up steps	–	$50K$
Optimizer	Adam	Adam
Adam betas	$(0.8, 0.998)$	$(0.8, 0.998)$
Learning rate	5×10^{-4}	7×10^{-4}
LR decay γ	0.95	0.95
LR decay period	2000	2000
Weight decay	0	0
Random seed	42	42

G.4 Robustness to Network Architecture

This section reports additional results obtained under the same experimental protocols as in the main experiments, but using lighter neural architectures for the acquisition policy, and smaller computational budgets, while retaining the downstream action architectures used in the main experiments. The goal is not to improve upon the main quantitative results, but rather to assess whether the qualitative conclusions are robust to reduced model capacity. In particular, we investigate whether the relative behaviour of ACTION-BED and the information-theoretic baselines DAD and IDAD remains consistent. Across tasks, the experimental setting, simulators, evaluation metrics, and train/test protocols are kept identical to those used in the corresponding main experiments. As expected, this generally leads to weaker absolute performance compared with the main results.

The precise architecture depends on the structure of the task. For exchangeable observation models, we use a DeepSets-style architecture in which each design–observation pair (ξ_s, y_s) is encoded independently and the resulting embeddings are aggregated by sum or mean pooling. For sequential and non-exchangeable simulators, such as the stochastic pendulum, the history is instead processed by a recurrent network, since the likelihood depends explicitly on the previous state. For each task, the downstream action policy π_a^ψ is kept identical to that used in the corresponding main experiment. It is implemented as an MLP taking the final history h_T as input and returning the estimate or action associated with the chosen downstream loss.

G.4.1 Source Location Finding

Training. For the source-location experiment, DAD and ACTION-BED use the same data-acquisition architecture, reported in Table 33. This architecture is the one used in Foster et al. [24] for the corresponding DAD experiment, ensuring that differences in performance are not driven by changes in policy-network capacity. Each design–observation pair (ξ, y) is first processed by a pair encoder, which maps it to a fixed-dimensional embedding. The embeddings associated with the history h_t are then aggregated by a permutation-invariant pooling operation, either sum or mean pooling, and passed to an emitter network that outputs the next design ξ_{t+1}. Both the pair encoder and the emitter are fully connected MLPs.

The RANDOM baseline does not use a learned acquisition network: designs are sampled independently from the initial prior distribution $p(\theta)$. The corresponding training hyperparameters for the learned policies and downstream action networks are reported in Table 34.

Table 35: **Source Location Finding, lightweight architectures**. Uncertainties are reported as ± 1 s.e. over 2048 evaluation rollouts. Compute time includes training both the design and downstream action networks.

Method	Performance ($\pm$ 1 s.e.)				Compute
	MSE (10^{-2}) $\downarrow$	QLL $\downarrow$	sPCE $\uparrow$	sNMC $\uparrow$	Time (min) $\downarrow$
Action-BED (MSE)	$\mathbf{2.6 \pm 0.2}$	-5.59 ± 0.03	9.14 ± 0.04	9.95 ± 0.07	97
Action-BED (QLL)	4.2 ± 0.4	$\mathbf{-6.39 \pm 0.04}$	10.36 ± 0.04	12.72 ± 0.03	102
DAD (MSE)	13.6 ± 0.7	-3.72 ± 0.04	$\mathbf{10.38 \pm 0.03}$	$\mathbf{12.91 \pm 0.03}$	283
DAD (QLL)	15.5 ± 1.2	-4.02 ± 0.04	$\mathbf{10.38 \pm 0.03}$	$\mathbf{12.91 \pm 0.03}$	285
Random	22.4 ± 0.8	-3.16 ± 0.04	8.27 ± 0.04	8.49 ± 0.05	**31**

Results. The results are reported in Table 35. After training, all policies are evaluated using $L = 100{,}000$ contrastive samples and a batch size of 2000 rollouts for the sequential PCE and NMC estimates. Downstream losses are computed on independent batches of 2000 rollouts. Uncertainties are reported as ± 1 standard error across rollout histories. We also report the total training time for each method, including both acquisition-policy and downstream-action-network training when applicable.

G.4.2 Training Stability over Training Seeds

To assess the stability of ACTION-BED, we repeated training across multiple random seeds and evaluated the resulting policies under the same protocol. We found that performance remained consistent across runs, with only moderate variation in downstream loss and EIG estimates. Notably, the standard errors across independently trained policies are of a similar order of magnitude to those obtained from rollout-level uncertainty estimates, suggesting that training-seed variability is not substantially larger than evaluation noise. This suggests that the joint optimisation of the design and action policies is not overly sensitive to random initialisation or stochastic training trajectories. In particular, the observed gains over EIG-based baselines in terms of downstream losses are not driven by a single favourable run, but persist across independently trained policies.

Table 36: **Training stability over random seeds on source location finding.** Uncertainties are reported as one standard error over independently trained policies with 15 different random seeds. For each trained policy, metrics are estimated using the same evaluation protocol.

Method	**MSE** (10^{-2}) $\downarrow$	**QLL** $\downarrow$	**sPCE** $\uparrow$	**sNMC** $\uparrow$
ACTION-BED (MSE)	2.7 ± 0.1	-5.54 ± 0.02	9.04 ± 0.04	9.79 ± 0.06
ACTION-BED (QLL)	3.9 ± 0.3	-6.27 ± 0.02	10.21 ± 0.07	12.67 ± 0.09
DAD (MSE)	14.1 ± 0.3	-3.78 ± 0.05	10.36 ± 0.07	12.79 ± 0.05
DAD (QLL)	15.7 ± 0.9	-3.96 ± 0.06	10.36 ± 0.07	12.79 ± 0.05
Random	21.2 ± 0.4	-3.19 ± 0.08	8.21 ± 0.02	8.41 ± 0.02

G.4.3 Dynamical Systems

Architecture. For both pendulum tasks, the acquisition network must account for the temporal dependence of the simulator. We therefore use the same sequential architecture for the stochastic single and double pendulum, inspired by Ivanova et al. [42]. Each design–state pair (ξ_t, x_t) is first embedded by an MLP encoder, and the resulting sequence of embeddings is processed by a two-layer LSTM. The final recurrent state is then passed through an emitter MLP, whose output dimension matches the design dimension of the task. The full acquisition architecture is summarised in Table 37.

The downstream action network is kept identical to that used in the corresponding main experiments. It is implemented as an MLP taking the full trajectory h_T as input and returning the parameter estimate associated with the chosen downstream loss. Unless otherwise stated, all optimisation hyperparameters are also inherited from the main experiments. The only change is the training budget: for the stochastic single pendulum, all policy, downstream, and joint networks are trained for 30K gradient steps; for the double pendulum, all corresponding networks are trained for 50K gradient steps. No downstream-only warm-up phase is used in either task.

The results are reported in Tables 38 and 39. They are obtained using the same evaluation protocol as in the main experiments, with downstream losses computed on independent rollout batches and information-theoretic metrics estimated using sequential PCE and NMC bounds.

Table 37: **Acquisition architecture for the pendulum tasks**. $\dim(\xi) = 1$ for the stochastic single pendulum and $\dim(\xi) = 2$ for the double pendulum. The $\tanh$ output is subsequently rescaled to the task-specific control domain.

Module	Layer	Description	Dimension / Activation
Pair encoder	Input	(ξ_t, x_t)	$\dim(\xi) + \dim(x)$ / –
	H1	Fully connected	256 / ReLU
	H2	Fully connected	256 / ReLU
	Output	Fully connected	64 / –
History aggregator	Input	$\{R(\xi_s, x_s)\}_{s=0}^{t}$	$(t+1) \times 64$ / –
	H1	LSTM	64 / –
	H2	LSTM	64 / –
Emitter	Input	Final recurrent state	64 / –
	H1	Fully connected	256 / ReLU
	H2	Fully connected	256 / ReLU
	Output	Design ξ_{t+1}	$\dim(\xi)$ / $\tanh$

Table 38: **Stochastic Single Pendulum**. Uncertainties are reported as ± 1 s.e over 2048 rollout histories.

Method	Performance ($\pm$ 1 s.e.)				Compute
	MSE (10^{-2}) ↓	QLL ↓	sPCE ↑	sNMC ↑	Time (min) ↓
Action-BED (MSE)	$\mathbf{1.0 \pm 0.09}$	-5.30 ± 0.03	$\mathbf{3.35 \pm 0.02}$	$\mathbf{3.40 \pm 0.02}$	45
Action-BED (QLL)	1.1 ± 0.11	$\mathbf{-5.85 \pm 0.04}$	2.55 ± 0.01	2.66 ± 0.01	47
DAD	1.7 ± 0.16	-5.34 ± 0.05	2.63 ± 0.12	2.65 ± 0.11	242
Random	2.90 ± 0.13	-4.15 ± 0.08	1.39 ± 0.07	1.42 ± 0.08	**5**

Table 39: **Stochastic Double Pendulum**. Uncertainties are reported as ± 1 s.e over 2048 rollout histories.

Method	Performance ($\pm$ 1 s.e.)				Compute
	MSE (10^{-3}) ↓	QLL ↓	sPCE ↑	sNMC ↑	Time (min) ↓
Action-BED (MSE)	$\mathbf{0.7 \pm 0.05}$	-9.25 ± 0.02	10.40 ± 0.02	14.50 ± 0.13	276
Action-BED (QLL)	1.4 ± 0.08	$\mathbf{-9.95 \pm 0.04}$	9.90 ± 0.03	12.50 ± 0.07	274
DAD (MSE)	1.0 ± 0.04	-9.32 ± 0.03	$\mathbf{10.75 \pm 0.03}$	$\mathbf{14.10 \pm 0.14}$	582
Random	2.0 ± 0.09	-6.80 ± 0.05	8.26 ± 0.06	8.41 ± 0.08	**58**

Table 40: **MNIST masked classification acquisition-policy architecture**.

Module	Layer	Description	Dimension / Activation
Pair encoder	Input	(ξ_t, y_t)	$\dim(\xi) + \dim(y)$ / –
	H1	Fully connected	256 / ReLU
	H2	Fully connected	128 / ReLU
	H3	Fully connected	64 / ReLU
	Output	Fully connected	16 / –
History aggregation	Input	$\{R(\xi_s, y_s)\}_{s=1}^{t}$	$t \times 16$ / –
	Pooling	Sum pooling	16 / –
Emitter	Input	Aggregated history	16 / –
	Output	Design ξ_{t+1}	2 / Sigmoid

Notes. The design-policy backbone is a DeepSets-style net. A sigmoid activation is used to constrain designs to $[0, 1]^2$ before reprojection onto the pixel grid.

G.4.4 MNIST Masked Classification

For the MNIST masked-classification task, we use the same class of acquisition architectures as in the source-location experiment. In particular, the design policies are implemented with a DeepSets-style backbone: each design–observation pair is encoded independently, the resulting embeddings are aggregated over the acquisition history, and an emitter network outputs the next design. The precise architecture is reported in Table 40.

The downstream classifier follows the same encoder–aggregation as summarised in Table 30. For iDAD, we use the same backbone for the design policy but the critic encoder is used as a downstream classifier as in the main experiment.

Training hyperparameters for all methods are kept identical to those used in the corresponding main experiment. The corresponding classification results, including cross-entropy loss and accuracy on both training and test sets, are presented in Table 41.

Table 41: **MNIST masked classification**. CE denotes cross-entropy loss and Acc. denotes classification accuracy. Metrics are computed after training on the full training and test datasets. Uncertainties are reported as one standard error across evaluation batches. Compute time includes training the acquisition policy and, when applicable, the downstream classifier.

Method	Performance ($\pm$ 1 s.e.)				Compute
	Train CE ↓	Train Acc. (%) ↑	Test CE ↓	Test Acc. (%) ↑	Time (min) ↓
Action-BED	$\mathbf{0.050 \pm 0.002}$	$\mathbf{98.24 \pm 0.04}$	$\mathbf{0.128 \pm 0.006}$	$\mathbf{96.55 \pm 0.13}$	**2.5**
DAD	0.398 ± 0.004	83.10 ± 0.03	0.609 ± 0.010	77.75 ± 0.12	18
iDAD	0.113 ± 0.003	96.28 ± 0.07	0.175 ± 0.007	94.76 ± 0.19	5
Random	0.786 ± 0.015	71.78 ± 0.87	0.797 ± 0.015	71.40 ± 0.86	5

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction make the following claims about our method: singly intractable, loss/task-driven, naturally implicit, easily customisable to downstream tasks, avoiding the need to explicit posterior or marginal likelihood estimation, improved performance on downstream tasks, often outperforming state-of-the-art in terms of EIG. These claims are all true and confirmed by the remainder of the paper. The abstract and introduction also state that the method requires reparameterisable simulations from the joint model over parameters and data, and the ability to evaluate the differentiable loss function. These are the main limitations of our work.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: In Section 4 we acknowledge that the approach requires either reparameterisable simulations of the observation model, or the ability to evaluate the gradient of the log-likelihood of the observation model. In Appendix C we detail the precise technical conditions under which the reparameterisation trick can be applied.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.

- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: Theorem 1 is stated in full with all the relevant conditions, and a complete proof is provided in Appendix B.1. In Appendix A we also provide a formal Proposition (1) and proof showing the validity of exchanging the minimisation over actions to a minimisation over action policies outside the expectation.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Our contribution is a new method algorithm which is fully described in Algorithm 1 and Algorithm 2. Furthermore, we provide detailed description of the architectures and training procedures used to obtain our results in Appendix E and Appendix G respectively.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example

(a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
(b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The code is publicly available, and a link to the GitHub repository is provided in the paper.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Comprehensive experimental details are provided in Appendix G.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: All metrics are reported with standard errors, which are appropriately stated in the table notes.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The GPU resources used are reported in Appendix G.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: Our research does not involve human participants or sensitive data. We are not aware of any potential negative societal impacts resulting from our work.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.

- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Examples of positive societal impacts are given in the text, while both positive and negative impacts are discussed in the conclusion.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The contributions of this paper do not directly have significant risks of misuse.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

Justification: The software libraries used in our experiments are properly cited.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The research does not involve crowdsourcing nor experiments with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLMs do not contribute to nor form part of any component of our proposed contributions.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.