BAYESIAN MODELING AND UNCERTAINTY QUANTIFICATION FOR LEARNING TO OPTIMIZE: WHAT, WHY, AND HOW

Anonymous authors
Paper under double-blind review

ABSTRACT

Optimizing an objective function with uncertainty awareness is well-known to improve the accuracy and confidence of optimization solutions. Meanwhile, another relevant but very different question remains yet open: how to model and quantify the uncertainty of an optimization algorithm (a.k.a., optimizer) itself? To close such a gap, the prerequisite is to consider the optimizers as sampled from a distribution, rather than a few prefabricated and fixed update rules. We first take the novel angle to consider the algorithmic space of optimizers, and provide definitions for the optimizer prior and likelihood, that intrinsically determine the posterior and therefore uncertainty. We then leverage the recent advance of learning to optimize (L2O) for the space parameterization, with the end-to-end training pipeline built via variational inference, referred as uncertainty-aware L2O (**UA-L2O**). Our study represents the first effort to recognize and quantify the uncertainty of the optimization algorithm. The extensive numerical results show that, UA-L2O achieves superior optimization performances in two out of three test functions, the loss function in data privacy attack, and the energy function in docking four out of five protein pairs, with the most accurate posterior estimation and uncertainty quantification. Besides, the optimized solutions close to the real solution are of a larger population and tighter confidence intervals, demonstrating a better calibration capability. All our codes will be publicly released upon acceptance.

1 INTRODUCTION

Computational models of many real-world applications involve optimizing non-convex objective functions. As the non-convex optimization problem is NP-hard, no optimization algorithm (or optimizer) could guarantee the global optima in general. Instead their solutions' usefulness (sometimes based on their proximity to optima), when the optima are unknown, can be very uncertain. Being able to quantify such uncertainty is important to not only assessing the solution uncertainty after optimization but also enhancing the search efficiency during optimization. For instance, reliable and trustworthy machine learning models demand uncertainty awareness and quantification (UQ) during training (optimizing) such models, whereas in reality deep neural networks without proper modelling of uncertainty suffer from overconfidence and miscalibration (Guo et al., 2017). In another example of 3D prediction of protein-protein interactions, even though there exists the model uncertainty of the objective function and the data uncertainty of the protein structure data (Cao & Shen, 2020), state-of-the-art methods only predict several ranked solutions (Porter et al., 2019) without any associated uncertainty, which is hard for biologists to interpret.

Despite progress in optimization with uncertainty-awareness, fundamental open questions remain: existing methods consider uncertainty either within the data or the model (including objective functions) (Kendall & Gal, 2017; Ortega et al., 2012; Cao & Shen, 2020), whereas inconspicuous attention was paid to the uncertainty arising from the **optimizer** that is directly responsible for deriving the end solutions. The optimizer is usually prefabricated and fixed. For instance, there are several popular update rules in Bayesian optimization, such as expected improvement (Vazquez & Bect, 2010) or upper confidence bound (Srinivas et al., 2009), that are chosen and unaltered for the entire process. For Bayesian neural networks training, the update rule of the iterative optimizer is usu-

ally chosen off-the-shelf like Adam, SGD, or RMSDrop. In principle, it is important to recognize the existence of an optimization algorithm **space** where the specific optimizer lies as well as the importance of the optimizer uncertainty, intrinsically defined in the space to the optimization and UQ solutions. In practice, such uncertainty is unwittingly omitted in the current context where an optimizer is treated as a *de facto* specified sample in the space.

A naïve characterization for a hand-built optimizer is to rely on a few of its hyper-parameters (Wenzel et al., 2020; Lorraine & Duvenaud, 2018), whereas such a parameterization space could be too biased and restricted to span a broader and more complicated optimizer manifold, potentially resulting in inaccurate UQ results (see Section 4, confidence estimation for test functions). Fortunately, the recent surge of learning to optimize (L2O) (Andrychowicz et al., 2016a; Li & Malik, 2016; Chen et al., 2021) which is another prominent optimization paradigm, parameterized by a neural network using high-dimensional weights and following the data-driven ethos to learn the update rule adaptively, therefore introducing the less inductive bias and accompanied with the universal approximation ability, indicates a plausible solution to depict the sophisticated and more complete optimizer space.

Contributions. The aforementioned introduction reveals and discusses the fundamental question of why modelling the optimizer uncertainty, and our contingent questions would be, what defines the optimizer uncertainty and how to enable UQ during optimization. We give the following answers. (i) We define the prior and likelihood of the optimizer (Neal, 2012), which routinely determine the optimizer posterior with the general product rule and Bayes' theorem. The optimizer uncertainty is therefore well-defined. (ii) To unseal UQ within optimization, we treat an optimizer as a random sample from the algorithmic space. We leverage the surging L2O technique (Andrychowicz et al., 2016a; Li & Malik, 2016) with high-dimensional weights for the optimizer space parameterization. (iii) We further construct the end-to-end training pipeline via variational inference (Kingma & Welling, 2013; Higgins et al., 2016) to avoid the expensive computational cost of Markov chain Monte Carlo (MCMC) and degenerated posterior. The newly trained L2O is referred as uncertainty-aware L2O (**UA-L2O**). To summary, the core intellectual value of this work is for the first time, we recognize and quantify a novel form of uncertainty that lies in the optimization algorithm space parameterized by L2O, apart from the classical data- or model-based uncertainties (also known as epistemic and aleatoric uncertainties (Fox & Ülkümen, 2011)).

Extensive experiments show that UA-L2O achieves the optimization performance, superior in two out of three test functions with high variable dimensions, best in the loss function of data privacy attack, and exceeded in the energy function of docking four out of five protein pairs. Besides, it estimated the most accurate posterior with better calibration capability, with the solutions closer to the ground truth of a larger population and tighter confidence intervals.

2 PRELIMINARIES

2.1 LEARNING TO OPTIMIZE

Let us consider an optimization problem: $\min_{\mathbf{x}} f(\mathbf{x})$ where $\mathbf{x} \in \mathbb{R}^d$. A classic optimizer often iteratively updates $\mathbf{x}$ based on a handcrafted rule. For example, the first-order gradient descent algorithm takes an update at iteration t based on the local gradient at the instantaneous point $\mathbf{x}_t$: $\mathbf{x}_{t+1} = \mathbf{x}_t - \alpha \nabla f(\mathbf{x}_t)$, where α is the step size.

Learning to Optimize (L2O) has lots of freedom to use the available information. We define $\mathbf{z}_t$ as the optimization historical information available up to time t, e.g., including the existing iterates $\mathbf{x}_0, \ldots, \mathbf{x}_t$, and/or their gradients $\nabla f(\mathbf{x}_0), \ldots, \nabla f(\mathbf{x}_t)$. L2O models an update rule by a predictor function g of $\mathbf{z}_t$: $\mathbf{x}_{t+1} = \mathbf{x}_t - g(\mathbf{z}_t; \boldsymbol{\phi})$, where the mapping of g is parameterized by $\boldsymbol{\phi}$. Finding an optimal update rule can be formulated mathematically as searching for a good $\boldsymbol{\phi}$ over the parameter space of g. Practically, g is often a neural network. Since neural networks are universal approximators, L2O has the potential to discover completely new update rules without relying on existing rules. In order to find a desired $\boldsymbol{\phi}$ associated with a fast optimizer, (Andrychowicz et al., 2016b) proposed to minimize the weighted sum of the objective function $f(\mathbf{x}_t)$ over a time span T:

$$\min_{\boldsymbol{\phi}} \sum_{t=1}^{T} w_t f(\mathbf{x}_t), \quad \text{with} \quad \mathbf{x}_{t+1} = \mathbf{x}_t - g(\mathbf{z}_t; \boldsymbol{\phi}),\ t = 0, \ldots, T-1, \tag{1}$$

where $w_1, \ldots, w_T$ are the weights whose choices vary from case to case and depend on empirical settings. Note that $\boldsymbol{\phi}$ determines the objective value through determining the iterates $\mathbf{x}_t$. L2O solves the problem (1) for a desirable $\boldsymbol{\phi}$ and correspondingly the update rule $g(\mathbf{z}_t; \boldsymbol{\phi})$.

A typical L2O workflow is divided into two stages (Chen et al., 2021): a meta-training stage that learns the optimizer with a set of similar optimizees from the task distribution; and a meta-testing stage that applies the learned optimizer to training new unseen optimizees. The meta-training process often occurs offline and is time consuming. However, the online application of the method is (aimed to be) time saving.

2.2 The Uncertainty of Optimization

The choice of optimizers is recognized to remarkably impact the solution quality, especially for non-convex and rugged objectives such as various loss functions for training deep networks. While non-convex optimization is NP-hard and the global optimum is never guaranteed, domain users usually can only blindly take the solutions given by a particular optimizer, or choose from several well-known algorithms based on human experiences or on an *ad hoc* basis. Such trial-and-error selections are biased and restricted to span a more complete optimizer manifold, and computationally expensive, even intractable when new algorithms are to be discovered by L2O, whose "fitness" or confidence on certain problem instances (especially instances with shifts from the target task distributions) is never known. Given a specific problem instance, if we can provide a confidence score or quantify the uncertainty of each candidate optimizer, it will certainly improve the selection and calibration of optimization algorithms, in particular the learned optimizers since they are essentially all "new", and hence enhancing the reliability of L2O.

As for non-convex objectives, different choices of optimizers, even each algorithm's hyper-parameter or initialization variations, can lead to vastly different solutions, implying a new type of uncertainty arising from choosing optimizers. Stochastic optimization methods like random search (Zhigljavsky, 2012), simulated annealing (Kirkpatrick et al., 1983) and genetic algorithms (Goldenberg, 1989) inject the randomness into algorithms to reduce solution uncertainty. However, they cannot reliably quantify its effect on the final solutions. Several works explored uncertainty quantification during optimization, by Bayesian optimization (Hennig & Schuler, 2012; Wang & Jegelka, 2017) or Monte Carlo sampling (Ortega et al., 2012). However, those methods are often computationally too expensive, not to mention applied within the already costly L2O process. More importantly, they neither explicitly consider the uncertainty within the optimizer itself, nor concern the automatic learning and discovery of the most suitable optimizer.

3 Technical Approach

To quantify and address such "fitness" especially for the L2O parameterization in a principled way, we will introduce this new **optimizer uncertainty** arising from the choice of optimization algorithms, which is different from the classical data- or model-based uncertainties (also known as epistemic and aleatoric uncertainties (Fox & Ülkümen, 2011)). We will explore: how to quantify optimizer uncertainty? how to design a learned optimizer so that its uncertainty can be more accurately quantified? what is the best calibrated optimizer we can choose or learn for a given problem?

The core innovation is to treat an optimizer as a random sample from a continuous algorithmic space, rather than one of a few hand-crafted update rules, so as to model the intrinsic uncertainty within the optimizer. This novel view of algorithmic space is particularly enabled by L2O. Note an L2O update rule is typically parameterized by a neural network g, with its inputs $\mathbf{z}_t$ and learnable parameters $\boldsymbol{\phi}$. Its learning capacity allows us to "sample" optimizers by taking $g(\cdot\,; \boldsymbol{\phi})$ with different weights $\boldsymbol{\phi}$. That is sufficiently versatile and flexible in practice.

3.1 Concepts for Optimizer Uncertainty

Without loss of generality, we define am iterative algorithmic space $\mathcal{G}$, where each point $g \in \mathcal{G}$ produces an update $g(\mathbf{z}_t)$ dependent on the current/past zero-th order and/or first-order information. We treat g as a random vector from $\mathcal{G}$, that leads to defining the optimizer uncertainty as follows:

Definition 1 (Optimizer Uncertainty) *Let $\mathcal{G}$ be the algorithmic space, where each point $g \in \mathcal{G}$ is an optimizer (omitting $\boldsymbol{\phi}$ parameterization). We assume that*

1. *g has a prior distribution $p(g)$;*
2. *Its likelihood can be interpreted as $p(\mathbf{z}_t|\mathbf{z}_{t_0}, g) = \prod_{i=t_0+1}^{t} p(\mathbf{x}_i|\mathbf{z}_{i-1}, g)$, $\forall t_0 < t$.*

The likelihood factorization is based on causality of first-order optimizers, i.e., $(\mathbf{x}_i, \mathbf{z}_i)$ depends solely on $\{(\mathbf{x}_{i_0}, \mathbf{z}_{i_0})\}_{i_0<i}$ given g. Thereby, we define the optimizer uncertainty at step t as the posterior of g, that is a conditional distribution on the t steps of the trajectory, using Bayes' theorem:

$$p(g|\mathbf{z}_t) \propto p(g) \prod_{i=1}^{t} p(\mathbf{x}_i|\mathbf{z}_{i-1}, g). \tag{2}$$

Prior arts on hyper-parameter optimization (HPO) for classical algorithms (Feurer & Hutter, 2019) could be viewed as a special case of L2O (e.g., only a few hyper-parameters are "learnable", which resulting in a much biased and restricted prior $p(g)$). The seminal work Bergstra et al. (2011) optimized hyper-parameters using posterior-based fitness modeling. Recently, Wenzel et al. (2020) proposed a non-Bayesian approach of hyper-parameter ensembling that can estimate the predictive uncertainty of an optimization algorithm by varying hyper-parameters. Our work generalizes from tuning a few hyper-parameters to learning the entire optimizer space, and extends a Bayesian treatment. Since L2O involves much higher-dimensional parameters compared to HPO, its uncertainty quantification calls for computationally tractable and scalable approaches, which we will discuss next.

3.2 L2O WITH BAYESIAN UNCERTAINTY QUANTIFICATION

Now that we have the optimizer space $\mathcal{G}$ parameterized in $\boldsymbol{\phi}$, an optimizer g is modeled as a random vector in the space. That allows for reducing the modeling of optimizer uncertainty $p(g|\mathbf{z}_t)$ to modeling the posterior $p(\boldsymbol{\phi}|\mathbf{z}_t)$. For simplicity, we use a Gaussian prior of $\boldsymbol{\phi}$: $p(\boldsymbol{\phi}) \propto \exp(-\frac{1}{2\lambda}\|\boldsymbol{\phi}\|_2^2)$ where λ is a constant. We also follow the widely-adopted idea in (Gal & Ghahramani, 2016; Bishop & Tipping, 2003; Ortega et al., 2012; Cao & Shen, 2020), to view the loss function value as proportional to the negative logarithmic likelihood, i.e., $p(\mathbf{x}_t|\mathbf{z}_{t-1}, g) = p(\mathbf{x}_t|\mathbf{z}_{t-1}, \boldsymbol{\phi}) \propto \exp(-f(\mathbf{x}_t^{\boldsymbol{\phi}}))$. We can then calculate the posterior based upon equation (2):

$$p(\boldsymbol{\phi}|\mathbf{z}_T) \propto \exp(-\lambda\|\boldsymbol{\phi}\|_2^2) \prod_{t=1}^{T} \exp(-f(\mathbf{x}_t^{\boldsymbol{\phi}})) = \exp\Big(-\sum_{t=1}^{T} f(\mathbf{x}_t^{\boldsymbol{\phi}}) - \frac{1}{2\lambda}\|\boldsymbol{\phi}\|_2^2\Big). \tag{3}$$

Directly optimizing the posterior via Markov chain Monte Carlo (MCMC) encounters the computational intractability for the high-dimensional weights in L2O, and leads to a degenerated point estimation without uncertainty. Instead, we introduce a variational distribution $q(\boldsymbol{\phi}; \boldsymbol{\theta})$ (Kingma & Welling, 2013) with the variational parameter $\boldsymbol{\theta}$, yielding the evidence lower bound (ELBO):

$$\text{ELBO} = -\text{KL}[q(\boldsymbol{\phi}; \boldsymbol{\theta})||p(\boldsymbol{\phi}|\mathbf{z}_T)] = -\mathbb{E}_{\boldsymbol{\phi}\sim q(\boldsymbol{\phi};\boldsymbol{\theta})} \sum_{t=1}^{T} f(\mathbf{x}_t^{\boldsymbol{\phi}}) - \text{KL}[q(\boldsymbol{\phi}; \boldsymbol{\theta})||p(\boldsymbol{\phi})]. \tag{4}$$

We refer to the L2O under the new meta-training loss (4) w.r.t $\boldsymbol{\theta}$ as uncertainty-aware L2O (**UA-L2O**), which can be trained end-to-end with the Gaussian parametrization and the reparametrization trick (Kingma & Welling, 2013; Higgins et al., 2016). The feed-forward computational chart of UA-L2O is depicted in Figure 1.

With the well-trained UA-L2O and the found $\boldsymbol{\theta}^*$, we ideally assume the function values of the trajectory are submartingales that $\mathbb{E}_{\boldsymbol{\phi}\sim q(\boldsymbol{\phi};\boldsymbol{\theta}^*)} f(\mathbf{x}_{t+1}^{\boldsymbol{\phi}}) \leq f(\mathbf{x}_t)$, $\mathbb{E}_{\boldsymbol{\phi}\sim q(\boldsymbol{\phi};\boldsymbol{\theta}^*)}|f(\mathbf{x}_t^{\boldsymbol{\phi}})| < \infty$. Then based upon martingale convergence theorem (Doob, 1953) $\lim_{t\to\infty} f(\mathbf{x}_t) = f(\mathbf{x}_\infty)$ exists almost surely, i.e. it converges with infinite optimization iterations. Within the finite steps, UA-L2O has a well-defined approximated posterior $q(\boldsymbol{\phi}; \boldsymbol{\theta}^*)$, and hence the baked-in ability to quantify optimizer uncertainty at the solution. Therefore, the posterior of solutions is calculated by the integration with the variational distribution $q(\boldsymbol{\phi}; \boldsymbol{\theta}^*)$ as (which can be estimated by Monte Carlo sampling):

$$p(\mathbf{x}^*|\mathbf{z}_T) = \int p(\mathbf{x}^*|\mathbf{z}_T, \boldsymbol{\phi}) q(\boldsymbol{\phi}; \boldsymbol{\theta}^*) d\boldsymbol{\phi}. \tag{5}$$

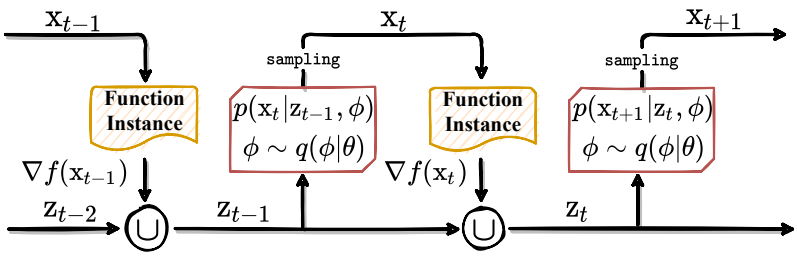

Figure 1: Feed-forward computational chart of UA-L2O. Notations are consistent with that in main text.

4 EXPERIMENTS

We examine our proposed UA-L2O on optimizing: (i) non-convex test functions, (ii) loss functions in data privacy attack, and (iii) energy functions in predicting 3D protein-protein interactions (protein docking). We compare UA-L2O with three non-Bayesian methods Adam (Kingma & Ba, 2014), particle swarm optimization (PSO) (Kennedy & Eberhart, 1995), DM_LSTM (Andrychowicz et al., 2016a), and a Bayesian method BAL (Cao & Shen, 2020). All algorithms are running for 10,000 times with random initialization to obtain the empirical posterior distributions, and we access the optimization uncertainty with the following expression (Ortega et al., 2012; Cao & Shen, 2020):

$$p\Big(\mathrm{lb}_\sigma \leq \|\mathbf{x}^* - \hat{\mathbf{x}}\|_2 \leq \mathrm{ub}_\sigma\Big) = 1 - \sigma, \tag{6}$$

where lb_σ and ub_σ are the lower and upper bounds at the confidence level of σ, and we choose the one with the lowest function value to be the estimated solution $\hat{\mathbf{x}}$. For the choice of assessing the proximity of solutions rather than function values $|f(\mathbf{x}^*) - f(\hat{\mathbf{x}})|$, the reason is in many optimization applications (e.g. privacy attack and protein docking), the objective function actually plays the role of the "tool" instead of the "goal", that the function value is not the key metric to assess the quality of solutions in particular when the objective function is non-convex and noisy.

Evaluation metrics. For optimization performance, we assess the expected distance between the optimized solutions and the global optimum $\mathbb{E}_{\mathbf{x}^* \sim p(\mathbf{x}^*)}\|\mathbf{x}^* - \mathbf{x}_{\text{true}}\|_2$. The lower the distance indicates the better the solution quality. For uncertainty quantification (UQ), we follow a traditional manner (Ortega et al., 2012; Cao & Shen, 2020) that, given a fixed confidence level σ (we take 0.8 and 0.9 in this paper), we assay the tightness of confidence interval $r_\sigma = \mathrm{ub}_\sigma - \mathrm{lb}_\sigma$ and the real confidence ϵ_σ. The real confidence ϵ_σ is defined by the fraction of sampled solutions that fall in the bounded region. A schematic illustration of the uncertainty assessment is shown in Figure 2.

Two-stage training. Due to the extremely high-dimensional optimizer space and the rugged posterior landscape, it is over-challenging to directly train the model through the ELBO in Equation (4). We therefore perform two-stage training, that first train our model in a non-Bayesian way and then use $\boldsymbol{\theta}$ as the warm start for the mean variational parameters of the second stage.

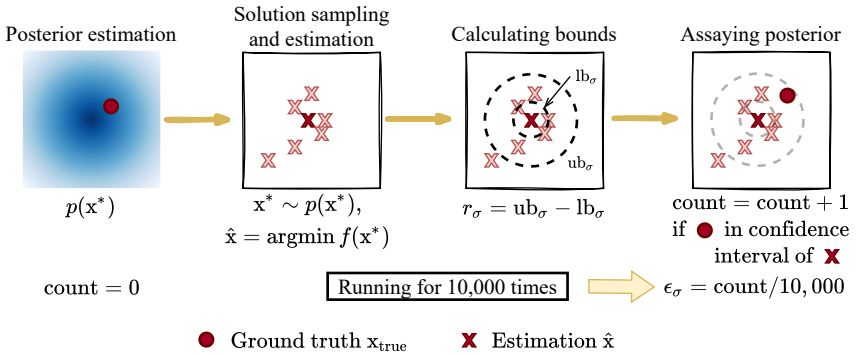

Figure 2: Computational procedure for uncertainty quantification: given a designated confidence score σ, calculating the length of confidence interval r_σ and its corresponding estimated confidence ϵ_σ.

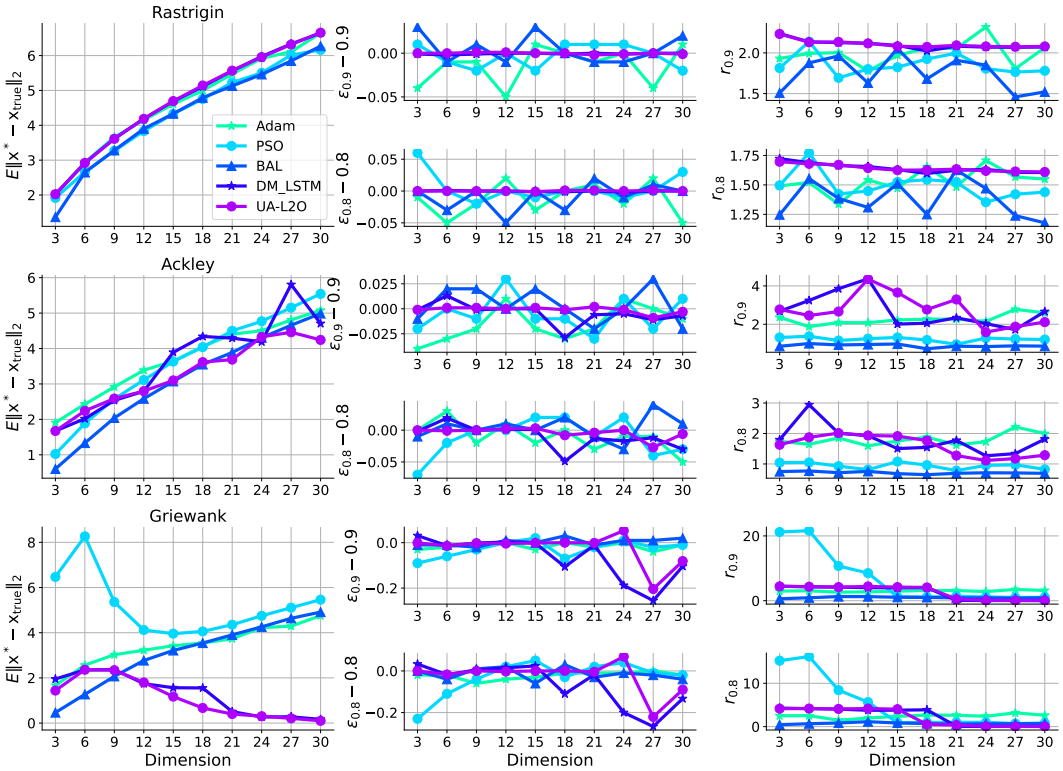

Figure 3: Optimization and uncertainty performance of different methods in three non-convex test functions. Different row represents different functions, and each column stands for: (i) 1st column is optimization performance, lower the better; (ii) 2nd column is precision of the estimated confidence, lower denoting more accurate posterior estimation; (iii) 3rd is the length of the confidence interval: only when with similar precision of the estimated confidence, lower indicating more certain solutions.

Implementations. The model is implemented in Tensorflow 1.13 (Abadi et al., 2016) and optimized by Adam (Kingma & Ba, 2014). For the L2O architecture, we use the coordinate-wise LSTM from (Andrychowicz et al., 2016a) containing 10,282 free parameters. For all experiments, the length of LSTM is set to be 20, with 5,000 training epochs for both training stages. We tune the λ values in equations (3) and (4) from $\{0.1,\ 1,\ 10\}$, validated by validation performance in test functions and the tightness of confidence intervals in privacy attack and protein docking. The ablation study on λ is presented in Appendix B showing that UA-L2O is not sensitive to its values.

4.1 Non-Convex Test Functions

Setup. We first evaluate the performance on the test functions in the global optimization benchmark (Jamil & Yang, 2013). We choose three extremely rugged, non-convex functions, Rastrigin, Ackley and Griewank with analytical forms shown in Appendix B, paired with dimensionalities in $\{3, 6, ..., 30\}$.

Results. We compare UA-L2O with competitors in optimization and UQ performances shown in Figure 3. In optimization performance (1st column), UA-L2O shows advantage in Ackley and Griewank functions when the dimensionality is above 12, while being slightly worse in Rastrigin. For the precision of posterior estimation (2nd column), UA-L2O performs the best in all but few cases for all three functions, yielding the most accurate confidence estimate, albeit with a looser confidence interval (3rd column). Note that other methods, when having tighter confidence intervals, were often too overconfident to realize that the true global optimum were outside such intervals.

The above results again reflect the importance of uncertainty-awareness in optimizers: when they obtain solutions without knowledge of optima, it would be helpful to have a secondary measurement to indicate the solution quality. In this way, the accurate posterior of UA-L2O (reflected in the 2nd column in Figure 3) ensures a good calibration potential between optimization performance (1st column) and the length of confidence interval (3rd column) in high-dimensional cases: better

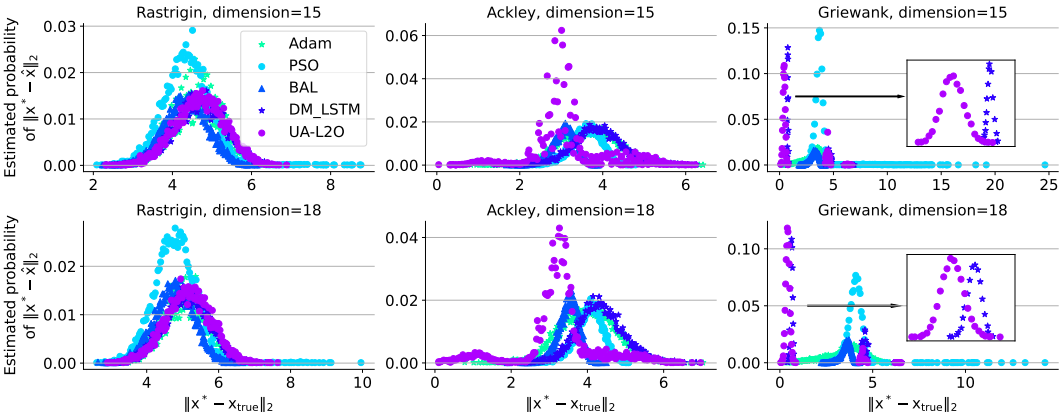

Figure 4: Scatter plot of confidence for solution regions segmented by $\|\mathbf{x}^* - \hat{\mathbf{x}}\|_2$, versus the corresponding optimization performance $\|\mathbf{x}^* - \mathbf{x}_{\text{true}}\|_2$ in three test functions. The peak closer to left side indicates the solution populations are in high confidence if they are closer to the optima.

optimization performance corresponds to tighter confidence interval. To further demonstrate the calibration capability of UA-L2O, we plot the confidence for solution regions segmented in $\|\mathbf{x}^* - \hat{\mathbf{x}}\|_2$ versus the corresponding optimization performance $\|\mathbf{x}^* - \mathbf{x}_{\text{true}}\|_2$ (see Appendix A for detailed procedure) in Figure 4. We find that when UA-L2O achieves the superior performance (in Ackley and Griewank), it generates solutions with higher confidence in the regions close to global optima and curves with the slimmer shape (to be more certain); otherwise the curve is flat (to be uncertain in the difficult function as Rastrigin with relatively worse solutions). In contrast the competitors do not show as clear trends as UA-L2O does: DM_LSTM does not distinguish between Rastrigin and Ackley and others barely show the trend.

4.2 Data Privacy Attack

Setup. We next apply our model to an application that critically needs UQ. As many machine learning models are deployed publicly, it is important to avoid leaking private and sensitive information, such as financial data and health data. Data privacy attack (Nasr et al., 2018) studies this problem by playing the role of hackers and attacking machine learning models to quantify the risk of privacy leakage. Better attacks would help models to be better prepared for privacy defense.

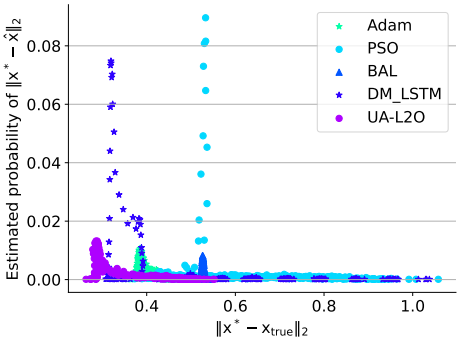

Figure 5: Scatter plot of confidence for solution regions segmented by $\|\mathbf{x}^* - \hat{\mathbf{x}}\|_2$, versus the corresponding optimization performance $\|\mathbf{x}^* - \mathbf{x}_{\text{true}}\|_2$ in data privacy attack.

We use the model and dataset in (Cao et al., 2019), where each input has 9 features involving patient genetic information shown in Appendix B, and the output is the probability of the clinical significance (having cancer or not) for a patient. We study the following model inversion attack (Fredrikson et al., 2015): given 5 features $\mathbf{x}' \in [0,1]^5$ out of 9 and the label y of each patient, the privacy attack aims to recover the rest 4 features $\mathbf{x}_{\text{true}} \in [0,1]^4$ (potentially sensitive patient information). Therefore, the optimization is $\min_{\mathbf{x}\in[0,1]^4} \sum (g(\mathbf{x}', \mathbf{x}) - y)^2$ for all patients where g is a trained predictor. The closeness between the optimized and real input features can quantify the risk of information leakage and the quality of the attack. We evaluate our method for all test cases in (Cao et al., 2019).

Results. We assess the optimization and UQ performances of UA-L2O as shown in Table 1. In optimization, UA-L2O outperforms all competitors, which again demonstrate the benefit of uncertainty-awareness. The UQ performance (the precision of estimated posterior) also stands out against competitors, although not as dominant as it does to competitors in test functions. We argue that, unlike pure optimization problems in test functions, the optimization for privacy attack involves the generalization issue in the machine learning paradigm: less generalizable samples pose challenges to not

Table 1: Optimization and uncertainty performance of different methods in data privacy attack.

Method	$\mathbb{E}\|\mathbf{x}^* - \mathbf{x}_{\text{true}}\|_2$	$\|\epsilon_{0.9} - 0.9\|$	$\|\epsilon_{0.8} - 0.8\|$	$r_{0.9}$	$r_{0.8}$
Adam	0.39	0.90	0.80	0.08	0.06
PSO	0.54	0.08	0.10	0.54	0.34
BAL	0.52	0.90	0.80	0.01	0.01
DM_LSTM	0.34	0.09	0.77	0.09	0.02
UA-L2O	**0.30**	0.25	0.27	0.06	0.04

only optimization but also UQ. To verify this argument, we plot optimization performance for individual samples versus their estimated posterior precision in Figure 6 showing a positive correlation: samples easier to generalize are closer to the optima (more leftwad in the x-axis), and more precise in estimated confidence (more downward in the y-axis).

In addition, for UA-L2O, the tightness of confidence interval is also well correlated with solution quality, indicating that it is more certain to the solutions closer to optima as shown in Figure 6. Similar to that in Section 4.1, we also plot confidence versus optimization performance in Figure 5, to further demonstrate the calibration capability of UA-L2O. Such uncertainty-awareness intuitively provides benefit for optimization performance as highlighted in Table 1.

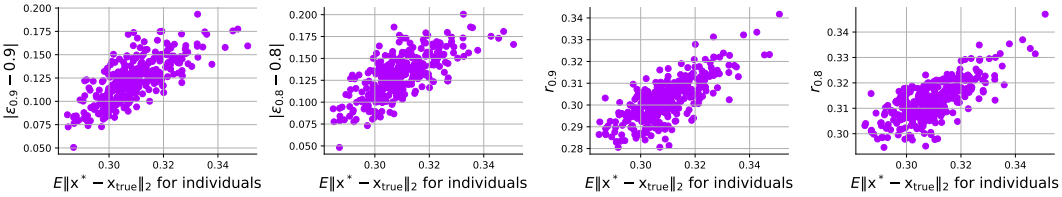

Figure 6: Optimization performance versus UQ results ($\epsilon_{0.9}, \epsilon_{0.8}, r_{0.9}$ and $r_{0.8}$) of UA-L2O for 318 test samples in data privacy attack.

4.3 Energy Functions for Protein Docking

Setup. We lastly examine our method using a bioinformatics application: predicting the 3D structures of protein-complexes (Smith & Sternberg, 2002), known as protein docking. *Ab initio* protein docking can be recast as optimizing a noisy and expensive energy function in a high-dimensional conformational space (Cao & Shen, 2020). While solving such optimization problems still remains difficult, quantifying the uncertainty of resulting optima (docking solutions) is even more challenging. The compared state-of-the-art method is BAL (Cao & Shen, 2020).

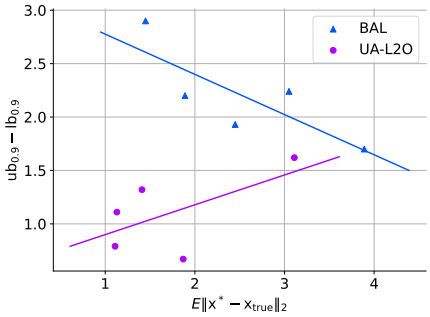

Figure 7: Scatter plot of optimization performance versus the length of confidence interval for five cases in protein docking. The Pearson correlation coefficient achieved by BAL is -0.80, and that by UA-L2O is **0.59**.

We calculate the energy function (objective function $f(\mathbf{x})$) in a CHARMM19 force field as in (Moal & Bates, 2010). 25 protein-protein complexes are chosen from the protein docking benchmark set 4.0 (Hwang et al., 2010) as the training set, which is shown in Appendix C. For each target, we choose 5 starting points (top-5 models from ZDOCK (Pierce et al., 2014)). In total, our training set includes 125 samples. Moreover, we parameterize the search space as $\mathbb{R}^{12}$ as in BAL. The resulting $f(\mathbf{x})$ is fully differentiable in the search space. We only consider 100 interface atoms due to the computational concern.

Results. We examine UA-L2O with numerical evidence shown in Table 2. In optimization, UA-L2O outperforms in four out of five cases, again demonstrating the advantage of the proposed algorithm. In UQ, UA-L2O not only achieves a more accurate posterior estimation as shown in Figure 8 and Appendix C, that three of five cases own ground truths located in 90% confidence interval, but also has a better calibration capability that the better solutions correspond to tighter intervals (to be more certain) as plotted in Figure 7 (with Pearson correlation coefficient of 0.59 versus -0.8 of BAL). The results show significant advantages of UA-L2O compared with the state of the art BAL in both optimization and UQ for protein docking.

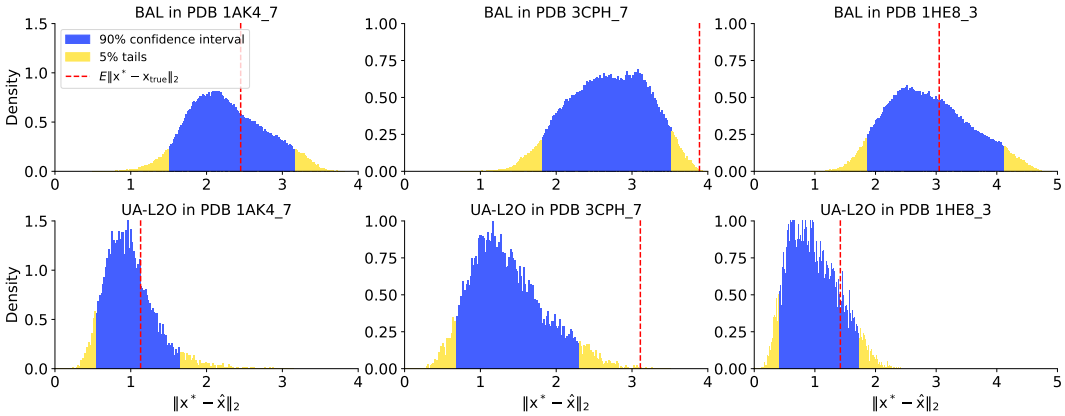

Figure 8: Estimated posterior distributions, confidence intervals and ground truth solutions for cases 1AK4_7, 3CPH_7 and 1HE8_3 in protein docking.

Table 2: Optimization and uncertainty performance of BAL and UA-L2O in protein docking.

Target PDB model (docking difficulty)	$\mathbb{E}\|\mathbf{x}^* - \mathbf{x}_{\text{true}}\|_2$ (Å) **BAL**	**UA-L2O**	$\mathbb{E}\|\mathbf{x}^* - \mathbf{x}_{\text{true}}\|_2 \in [\text{lb}_{0.9}, \text{ub}_{0.9}]$? **BAL**	**UA-L2O**	$\text{ub}_{0.9} - \text{lb}_{0.9}$ (Å) **BAL**	**UA-L2O**
1AHW_3 (easy)	1.89	**1.11**	No	Yes	2.20	0.79
1AK4_7 (easy)	2.45	**1.13**	Yes	Yes	1.93	1.11
3CPH_7 (medium)	3.89	**3.11**	No	No	1.70	1.62
1HE8_3 (medium)	3.05	**1.42**	Yes	Yes	2.24	1.32
1JMO_4 (difficult)	**1.45**	1.87	No	No	2.90	0.67

4.4 Result Summary

We briefly summarize the results as follows.

- In general, UA-L2O achieves better **optimization performance** with a few exceptions. This echos the hypothesis that the assessment of uncertainty is able to enhance the search efficiency and effectiveness during optimization.
- UA-L2O provides the most accurate **posterior estimation** most of the time. This empirical evidence verifies the effectiveness of our uncertainty-aware L2O inspired from theory.
- Accurate posterior estimation is a key for UA-L2O's superior **calibration capability**, such that UA-L2O generates solutions closer to optima with higher confidence and tighter confidence intervals, whereas competitors suffer from overconfidence or miscalibration. Such calibration capability is critical to detect the failure cases in real-world applications.

5 Conclusions

Current optimization algorithms, even with uncertainty-awareness, do not address the uncertainty arising within the optimizer itself. To close the gap, in this paper we attempt to ask and address three fundamental questions, why modelling the optimizer uncertainty, what defines the optimizer uncertainty, and how to enabling UQ during optimization. We first emphasize the uncertainty arising from the optimizer is a crucial source that directly responses for the end solutions deriving, except for data- and model-uncertainty. Next, we define the prior and likelihood of the optimizer which determines the optimizer posterior and the optimizer uncertainty. We further leverage learning to optimize (L2O) for the optimizer parameterization, treating an optimizer as a random sample from the algorithmic space, with the end-to-end training pipeline built via variational inference. Extensive experiments show UA-L2O achieves the optimization performance, superior in two out of three test functions with the high variable dimension, best in the loss function in data privacy attack, and exceeded in the energy function for protein docking in four out of five cases. Besides, it delivers the most accurate posterior estimation and calibration capability, with the solutions closer to the ground truth of a larger population and tighter confidence intervals. Our study represents the first effort to recognize and quantify the uncertainty of the optimization algorithm, with extensive support from numerical results and analysis.

REPRODUCIBILITY STATEMENT

To ensure reproducibility, we use the Machine Learning Reproducibility Checklist v2.0, Apr. 7 2020 (Pineau et al., 2021).

- For all **models** and **algorithms** presented,
  - **A clear description of the mathematical settings, algorithm, and/or model.** We clearly describe all of the settings, formulations, and algorithms in Section 3.
  - **A clear explanation of any assumptions.** We state our assumptions clearly in Section 3.
  - **An analysis of the complexity (time, space, sample size) of any algorithm.** We do not make the analysis.
- For any **theoretical claim**,
  - **A clear statement of the claim.** We provide a clear statement in Section 3.
  - **A complete proof of the claim.** We do not make theoretical proofs.
- For all **datasets** used, check if you include:
  - **The relevant statistics, such as number of examples.** We provide all the related statistics in Section 4 and Appendix B.
  - **The details of train/validation/test splits.** We give this information in our repository.
  - **An explanation of any data that were excluded, and all pre-processing step.** We give this information in our repository.
  - **A link to a downloadable version of the dataset or simulation environment.** Our repository contains all of the instructions to run experiments on the datasets in this work. We put it in supplementary materials.
  - **For new data collected, a complete description of the data collection process, such as instructions to annotators and methods for quality control.** We do not collect or release new datasets.
- For all shared **code** related to this work, check if you include:
  - **Specification of dependencies.** We give installation instructions in the README of our repository.
  - **Training code.** The training code is available in our repository.
  - **Evaluation code.** The evaluation code is available in our repository.
  - **(Pre-)trained model(s).** We do not release trained models.
  - **README file includes table of results accompanied by precise command to run to produce those results.** We include a README with detailed instructions to reproduce all of our experimental results.
- For all reported **experimental results**, check if you include:
  - **The range of hyper-parameters considered, method to select the best hyper-parameter configuration, and specification of all hyper-parameters used to generate results.** We provide all details of hyper-parameter tuning in Section 4.
  - **The exact number of training and evaluation runs.** Ten thousand independent repetitions are conducted for each experiments.
  - **A clear definition of the specific measure or statistics used to report results.** We provide all details of evaluation metrics in Section 4.
  - **A description of results with central tendency (e.g. mean) & variation (e.g. error bars).** We report confidence intervals for all experiments in Section 4.
  - **The average runtime for each result, or estimated energy cost.** We do not report the running time or energy cost.
  - **A description of the computing infrastructure used.** A clear description is presented in Section 3.

REFERENCES

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pp. 265–283, 2016.

Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in neural information processing systems*, pp. 3981–3989, 2016a.

Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in neural information processing systems*, pp. 3981–3989, 2016b.

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. *Advances in neural information processing systems*, 24, 2011.

Christopher M Bishop and Michael E Tipping. Bayesian regression and classification. *Nato Science Series sub Series III Computer And Systems Sciences*, 190:267–288, 2003.

Yue Cao and Yang Shen. Bayesian active learning for optimization and uncertainty quantification in protein docking. *Journal of chemical theory and computation*, 16(8):5334–5347, 2020.

Yue Cao, Yuanfei Sun, Mostafa Karimi, Haoran Chen, Oluwaseyi Moronfoye, and Yang Shen. Predicting pathogenicity of missense variants with weakly supervised regression. *Human mutation*, 40(9):1579–1592, 2019.

Tianlong Chen, Xiaohan Chen, Wuyang Chen, Howard Heaton, Jialin Liu, Zhangyang Wang, and Wotao Yin. Learning to optimize: A primer and a benchmark. *arXiv preprint arXiv:2103.12828*, 2021.

Joseph Leo Doob. *Stochastic processes*, volume 10. New York Wiley, 1953.

Matthias Feurer and Frank Hutter. Hyperparameter optimization. In *Automated Machine Learning*, pp. 3–33. Springer, Cham, 2019.

Craig R Fox and Gülden Ülkümen. Distinguishing two dimensions of uncertainty. *Perspectives on thinking, judging, and decision making*, 2011.

Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pp. 1322–1333, 2015.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pp. 1050–1059. PMLR, 2016.

David E Goldenberg. Genetic algorithms in search, optimization and machine learning, 1989.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1321–1330. JMLR. org, 2017.

Philipp Hennig and Christian J Schuler. Entropy search for information-efficient global optimization. *The Journal of Machine Learning Research*, 13(1):1809–1837, 2012.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. 2016.

Howook Hwang, Thom Vreven, Joël Janin, and Zhiping Weng. Protein-Protein Docking Benchmark Version 4.0. *Proteins*, 78(15):3111–3114, November 2010. ISSN 0887-3585. doi: 10.1002/prot.22830. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2958056/.

Momin Jamil and Xin-She Yang. A literature survey of benchmark functions for global optimisation problems. *International Journal of Mathematical Modelling and Numerical Optimisation*, 4(2): 150–194, 2013.

Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision?, 2017.

J Kennedy and R Eberhart. Particle swarm optimization, proceedings of ieee international conference on neural networks (icnn'95) in, 1995.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Scott Kirkpatrick, C Daniel Gelatt, and Mario P Vecchi. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.

Ke Li and Jitendra Malik. Learning to optimize. *arXiv preprint arXiv:1606.01885*, 2016.

Jonathan Lorraine and David Duvenaud. Stochastic hyperparameter optimization through hypernetworks. *arXiv preprint arXiv:1802.09419*, 2018.

Iain H. Moal and Paul A. Bates. SwarmDock and the Use of Normal Modes in Protein-Protein Docking. *International Journal of Molecular Sciences*, 11(10):3623–3648, September 2010. ISSN 1422-0067. doi: 10.3390/ijms11103623. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2996808/.

Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Stand-alone and federated learning under passive and active white-box inference attacks. *arXiv preprint arXiv:1812.00910*, 2018.

Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

Pedro Ortega, Jordi Grau-Moya, Tim Genewein, David Balduzzi, and Daniel Braun. A nonparametric conjugate prior distribution for the maximizing argument of a noisy function. In *Advances in Neural Information Processing Systems*, pp. 3005–3013, 2012.

Vikas Pejaver, Jorge Urresti, Jose Lugo-Martinez, Kymberleigh A Pagel, Guan Ning Lin, Hyun-Jun Nam, Matthew Mort, David N Cooper, Jonathan Sebat, Lilia M Iakoucheva, et al. Mutpred2: inferring the molecular and phenotypic impact of amino acid variants. *BioRxiv*, pp. 134981, 2017.

Brian G. Pierce, Kevin Wiehe, Howook Hwang, Bong-Hyun Kim, Thom Vreven, and Zhiping Weng. ZDOCK server: interactive docking prediction of protein–protein complexes and symmetric multimers. *Bioinformatics*, 30(12):1771–1773, 02 2014. ISSN 1367-4803. doi: 10.1093/bioinformatics/btu097. URL https://doi.org/10.1093/bioinformatics/btu097.

Joelle Pineau, Philippe Vincent-Lamarre, Koustuv Sinha, Vincent Lariviere, Alina Beygelzimer, Florence d'Alché Buc, Emily Fox, and Hugo Larochelle. Improving reproducibility in machine learning research. *Journal of Machine Learning Research*, 22:1–20, 2021.

K. A. Porter, I. Desta, D. Kozakov, and S. Vajda. What method to use for protein-protein docking? *Curr. Opin. Struct. Biol.*, 55:1–7, 04 2019.

Graham R Smith and Michael JE Sternberg. Prediction of protein–protein interactions by docking methods. *Current opinion in structural biology*, 12(1):28–35, 2002.

Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009.

Emmanuel Vazquez and Julien Bect. Convergence properties of the expected improvement algorithm with fixed mean and covariance functions. *Journal of Statistical Planning and inference*, 140(11):3088–3095, 2010.

Zi Wang and Stefanie Jegelka. Max-value entropy search for efficient bayesian optimization. In *International Conference on Machine Learning*, pp. 3627–3635, 2017.

Florian Wenzel, Jasper Snoek, Dustin Tran, and Rodolphe Jenatton. Hyperparameter ensembles for robustness and uncertainty quantification. *arXiv preprint arXiv:2006.13570*, 2020.

Anatoly A Zhigljavsky. *Theory of global random search*, volume 65. Springer Science & Business Media, 2012.

A CONFIDENCE ESTIMATION PROCEDURES

In evaluation, for the confidence versus optimization performance scatter plot, we first perform Monte Carlo sampling on the solution posterior, and then bin the solutions to estimate their fraction and corresponding average distance to the optimum. Detailed procedures can be found in Algorithms 1.

Algorithm 1: Confidence estimation with corresponding optimization performance

Input: Sampled solutions $\mathbf{x}^*$ with size N, estimated solution $\hat{\mathbf{x}}$, known optimum $\mathbf{x}_{\text{true}}$, bin width w.
Initialize: $S = \{\}$.
1. Calculate $\|\mathbf{x}^* - \hat{\mathbf{x}}\|_2$ for all sampled solutions, and bin them with w.
for each bin b **do**
  2. Calculate the population fraction $|b|/N$.
  3. Calculate average distance to $\mathbf{x}_{\text{true}}$: $\text{Mean}(\{\|\mathbf{x}^* - \mathbf{x}_{\text{true}}\|_2 | \mathbf{x}^* \in b\})$.
  4. $S \leftarrow S \cup (|b|/N, \text{Mean}(\{\|\mathbf{x}^* - \mathbf{x}_{\text{true}}\|_2 | \mathbf{x}^* \in b\}))$.
end for
Return: Scatter point set S.

B DATA DESCRIPTION

The analytical forms of three test functions are shown in Table 3. 9-dimension features of privacy attack data are generated from MutPred2 (Pejaver et al., 2017), which captures mutational impacts on protein structure, dynamics, and function, and grouped hierarchically into a custom oncology based on their inherent relationships. The list of features can be found in Table 4. For protein docking, the PDB IDs of training samples are shown in Table 5.

Table 3: The analytical forms of test functions $f_D(\mathbf{x})$ with the dimension D, and their corresponding flatness around global optima, estimated with the norm of Hessain matrix.

Function	Analytic form
Rastrigin	$f_D(\mathbf{x}) = \|\mathbf{x}\|_2^2 - \sum_{i=1}^{D} 10\cos(2\pi x_i) + 10D$
Ackley	$f_D(\mathbf{x}) = -20\exp(-0.2\sqrt{0.5\|\mathbf{x}\|_2^2}) - \sum_{i=1}^{D} \exp(\cos(2\pi x_i)/D) + \exp(1) + 20$
Griewank	$f_D(\mathbf{x}) = 1 + \sum_{i=1}^{D} \|\mathbf{x}\|_2^2/4000 - \prod_{i=1}^{D} \cos(x_i)$

Table 4: 9-dimension feature description for data privacy attack experiment.

Feature index	Property/molecular impact
1	Relative solvent accessibility
2	Allosteric site
3	Catalytic site
4	Secondary structure
5	Stability and conformation flexibility
6	Special structural signatures
7	Macromolecular binding
8	Metal binding
9	PTM site

Table 5: 4-letter ID of proteins used in protein docking training set.

Difficulty level	Protein Data Bank (PDB) code
Rigid	1N8O, 7CEI, 1DFJ, 1AVX, 1BVN, 1IQD, 1CGI, 1MAH, 1EZU, 1JPS, 1PPE, 1R0R, 2I25, 2B42, 1EAW, 2JEL, 1BJ1, 1KXQ, 1EWY
Medium	1XQS, 1M10, 1IJK, 1GRN
Flexible	1IBR, 1ATN

C More Evaluation Results

We provide more evaluation results as follows. (i) The ablation of λ values on three test functions is shown in Figure 9. (ii) The UQ results for test cases 1AHW_3 and 1JMO_4 in protein docking are plotted in Figure 10.

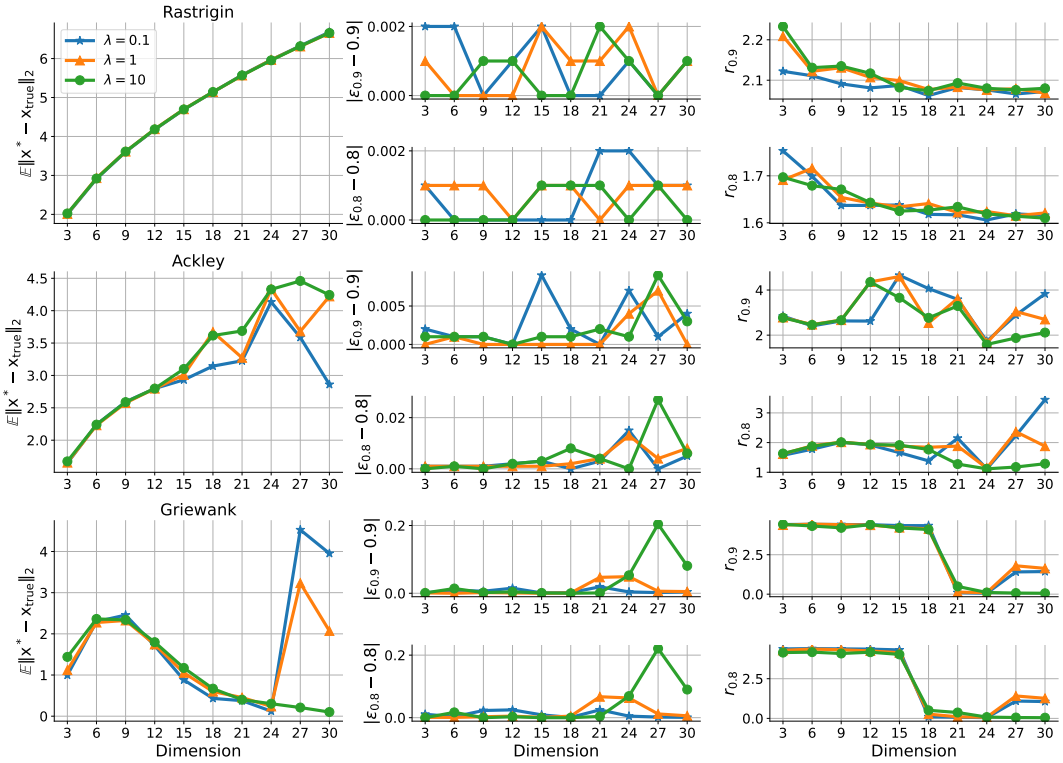

Figure 9: Optimization and uncertainty performance of UA-L2O with different λ in three non-convex test functions. Different row represents different functions, and each column stands for: (i) 1st column is optimization performance, lower the better; (ii) 2nd column is precision of the estimated confidence, lower denoting more accurate posterior estimation; (iii) 3rd is the length of the confidence interval: only when with similar precision of the estimated confidence would a tighter interval indicating more certain solutions.

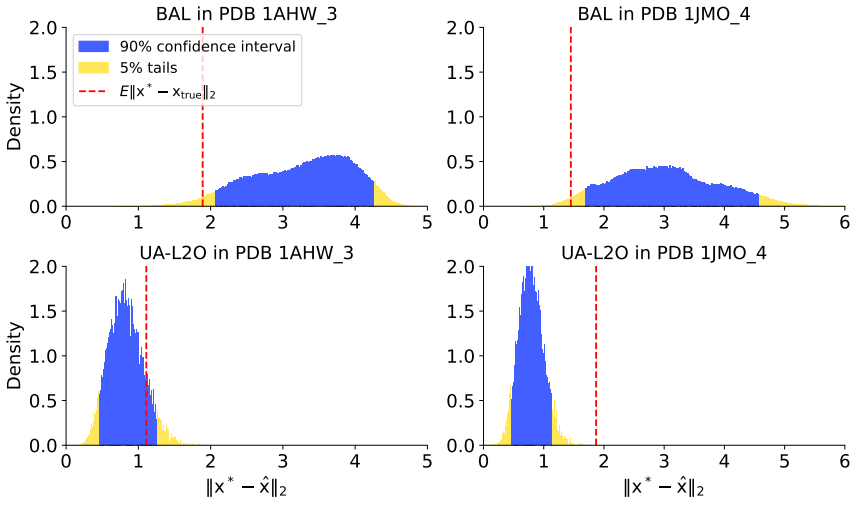

Figure 10: Estimated posterior distributions, confidence intervals and ground truth solutions for cases 1AHW_3 and 1JMO_4 in protein docking.