Batch Multi-Fidelity Active Learning with Budget Constraints

Shibo Li*, **Jeff M. Phillips***, **Xin Yu, Robert M. Kirby, and Shandian Zhe**
School of Computing, University of Utah
Salt Lake City, UT 84112
{shibo, jeffp, xiny, kirby, zhe}@cs.utah.edu

Abstract

Learning functions with high-dimensional outputs is critical in many applications, such as physical simulation and engineering design. However, collecting training examples for these applications is often costly, *e.g.*, by running numerical solvers. The recent work (Li et al., 2022) proposes the first multi-fidelity active learning approach for high-dimensional outputs, which can acquire examples at different fidelities to reduce the cost while improving the learning performance. However, this method only queries at one pair of fidelity and input at a time, and hence has a risk to bring in strongly correlated examples to reduce the learning efficiency. In this paper, we propose Batch Multi-Fidelity Active Learning with Budget Constraints (BMFAL-BC), which can promote the diversity of training examples to improve the benefit-cost ratio, while respecting a given budget constraint for batch queries. Hence, our method can be more practically useful. Specifically, we propose a novel batch acquisition function that measures the mutual information between a batch of multi-fidelity queries and the target function, so as to penalize highly correlated queries and encourages diversity. The optimization of the batch acquisition function is challenging in that it involves a combinatorial search over many fidelities while subject to the budget constraint. To address this challenge, we develop a weighted greedy algorithm that can sequentially identify each (fidelity, input) pair, while achieving a near $(1 - 1/e)$-approximation of the optimum. We show the advantage of our method in several computational physics and engineering applications.

1 Introduction

Applications, such as in computational physics and engineering design, often demand we calculate a complex mapping from low-dimensional inputs to high-dimensional outputs, such as finding the optimal material layout (output) given the design parameters (input), and solving the solution field on a mesh (output) given the PDE parameters (input). Computing these mappings is often very expensive, *e.g.*, iteratively running numerical solvers. Hence, learning a surrogate model to outright predict the mapping, which is much faster and cheaper, is of great practical interest and importance (Kennedy and O'Hagan, 2000; Conti and O'Hagan, 2010)

However, collecting training examples for the surrogate model becomes another bottleneck, since each example still requires a costly computation. To alleviate this issue, Li et al. (2022) developed DMFAL, a first deep multi-fidelity active learning algorithm, which can acquire examples at different fidelities to reduce the cost of data collection. Low-fidelity examples are cheap to compute (*e.g.*, with coarse meshes) yet inaccurate; high-fidelity examples are accurate but much more expensive (*e.g.*, calculated with dense grids). See Fig. 1 for an illustration. DMFAL uses an optimization-based acquisition method to dynamically identify the input and fidelity at which to query new examples, so as to improve the learning performance, lower the sample complexity, and reduce the cost.

*Equal contribution Correspondence to: Jeff M. Phillips, Shandian Zhe.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

Despite its effectiveness, DMFAL can only optimize and query at one pair of input and fidelity each time and hence ignores the correlation between consecutive queries. As a result, it has a risk of bringing in strongly correlated examples, which can restrict the learning efficiency and lead to a suboptimal benefit-cost ratio. In addition, the sequential querying and training strategy is difficult to utilize parallel computing resources that are common nowadays (*e.g.*, multi-core CPUs/GPUs and computer clusters) to query concurrently and to further speed up.

In this paper, we propose BMFAL-BC, a batch multi-fidelity active learning method with budget constraints. Our method can acquire a batch of multi-fidelity examples at a time to inhibit the example correlations, promote diversity so as to improve the learning efficiency and benefit-cost ratio. Our method can respect a given budget in issuing batch queries, hence are more widely applicable and practically useful. Specifically, we first propose a novel acquisition function, which measures the mutual information between a batch of multi-fidelity queries and the target function. The acquisition function not only can penalize highly correlated queries to encourage diversity, but also can be efficiently computed by an Monte-Carlo approximation. However, optimizing the acquisition function is challenging because it incurs a combinatorial search over fidelities and meanwhile needs to obey the constraint. To address this challenge, we develop a weighted greedy algorithm. We sequentially find one pair of fidelity and input each step, by maximizing the increment of the mutual information weighted by the cost. In this way, we avoid enumerating the fidelity combinations and greatly improve the efficiency. We prove that our greedy algorithm nearly achieves a $(1 - \frac{1}{e})$-approximation of the optimum, with a few minor caveats.

For evaluation, we examined BMFAL-BC in five real-world applications, including three benchmark tasks in physical simulation (solving Poisson's, Heat and viscous Burger's equations), a topology structure design problem, and a computational fluid dynamics (CFD) task to predict the velocity field of boundary-driven flows. We compared with the budget-aware version of DMFAL, single multi-fideity querying with our acquisition function, and several random querying strategies. Under the same budget constraint, our method consistently outperforms the competing methods throughout the learning process, often by a large margin.

2 Background

2.1 Problem Setting

Suppose we aim to learn a mapping $f : \Omega \subseteq \mathbb{R}^r \to \mathbb{R}^d$ where r is small but d is large, *e.g.*, hundreds of thousands. To economically learn this mapping, we collect training examples at M fidelities. Each fidelity m corresponds to mapping $f_m : \Omega \to \mathbb{R}^{d_m}$. The target mapping is computed at the highest fidelity, *i.e.*, $f(\mathbf{x}) = f_M(\mathbf{x})$. The other f_m can be viewed as a (rough) approximation of f. Note that d_m is unnecessarily the same as d for $m < M$. For example, solving PDEs on a coarse mesh will give a lower-dimensional output (on the mesh points). However, we can interpolate it to the d-dimensional space to match $f(\cdot)$ (this is standard in physical simulation (Zienkiewicz et al., 1977)). Denote by λ_m the cost of computing $f_m(\cdot)$ at fidelity m. We have $\lambda_1 \leq \ldots \leq \lambda_M$.

2.2 Deep Multi-Fidelity Active Learning (DMFAL)

To effectively estimate f while reducing the cost, Li et al. (2022) proposed DMFAL, a multi-fidelity deep active learning approach. Specifically, a neural network (NN) is introduced for each fidelity m, where a low-dimensional hidden output $\mathbf{h}_m(\mathbf{x})$ is first generated, and then projected to the high-dimensional observation space. Each NN is parameterized by $(\mathbf{A}_m, \mathbf{W}_m, \boldsymbol{\theta}_m)$, where $\mathbf{A}_m$ is the projection matrix, $\mathbf{W}_m$ is the weight matrix of the last layer, and $\boldsymbol{\theta}_m$ consists of the remaining NN parameters. The model is defined as follows,

$$\mathbf{x}_m = [\mathbf{x}; \mathbf{h}_{m-1}(\mathbf{x})], \;\; \mathbf{h}_m(\mathbf{x}) = \mathbf{W}_m \boldsymbol{\phi}_{\boldsymbol{\theta}_m}(\mathbf{x}_m), \;\; \mathbf{y}_m(\mathbf{x}) = \mathbf{A}_m \mathbf{h}_m(\mathbf{x}) + \boldsymbol{\xi}_m, \tag{1}$$

where $\mathbf{x}_m$ is the input to the NN at fidelity m, $\mathbf{y}_m(\mathbf{x})$ is the observed d_m dimensional output, $\boldsymbol{\xi}_m \sim \mathcal{N}(\cdot|\mathbf{0}, \tau_m \mathbf{I})$ is a random noise, $\boldsymbol{\phi}_{\boldsymbol{\theta}_m}(\mathbf{x}_m)$ is the output of the second last layer and can be viewed as a nonlinear feature transformation of $\mathbf{x}_m$. Since $\mathbf{x}_m$ includes not only the original input $\mathbf{x}$, but also the hidden output from the previous fidelity, *i.e.*, $\mathbf{h}_{m-1}(\mathbf{x})$, the model can propagate information throughout fidelities and capture the complex relationships (*e.g.*, nonlinear and nonstationary) between different fidelities. The whole model is visualized in Fig. 5 of Appendix. To estimate the posterior of the model, DMFAL uses a structural variational inference algorithm. A multi-variate Gaussian

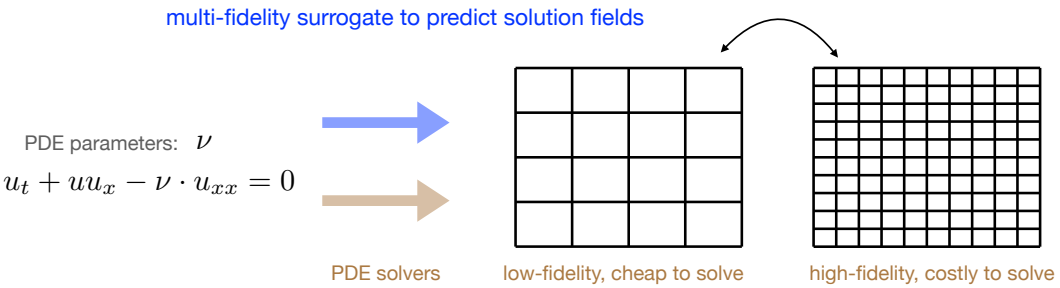


Figure 1: Illustration of the motivation and goal with physical simulation as an example. It is costly to solve every PDE from scratch. We aim to train a multi-fidelity surrogate model to directly predict high-fidelity solution fields given the PDE parameters. To further reduce the cost of collecting the training data with numerical solvers, we seek to develop multi-fidelity active learning algorithms.

posterior is introduced for each weight matrix, $q(\mathbf{W}_m) = \mathcal{N}\big(\mathrm{vec}(\mathbf{W}_m)|\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m\big)$. A variational evidence lower bound (ELBO) is maximized via stochastic optimization and the reparameterization trick (Kingma and Welling, 2013).

To conduct active learning, DMFAL views the most valuable example at each fidelity m as the one that can best help its prediction at the highest fidelity M (*i.e.*, the target function). Accordingly, the acquisition function is defined as

$$a(m, \mathbf{x}) = \frac{1}{\lambda_m}\mathbb{I}\big(\mathbf{y}_m(\mathbf{x}), \mathbf{y}_M(\mathbf{x})|\mathcal{D}\big) = \frac{1}{\lambda_m}\left(\mathbb{H}(\mathbf{y}_m|\mathcal{D}) + \mathbb{H}(\mathbf{y}_M|\mathcal{D}) - \mathbb{H}(\mathbf{y}_m, \mathbf{y}_M|\mathcal{D})\right), \qquad (2)$$

where $\mathbb{I}(\cdot,\cdot)$ is the mutual information, $\mathbb{H}(\cdot)$ is the entropy, and $\mathcal{D}$ is the current training dataset. The computation of the acquisition function is quite challenging because $\mathbf{y}_m$ and $\mathbf{y}_M$ are both high dimensional. To address this issue, DMFAL takes advantage of the fact that each low-dimensional output $\mathbf{h}_m(\mathbf{x})$ is a nonlinear function of the random weight matrices $\{\mathbf{W}_1, \ldots, \mathbf{W}_m\}$. Based on the variational posterior $\{q(\mathbf{W}_j)\}$, DMFAL uses the multi-variate delta method (Oehlert, 1992; Bickel and Doksum, 2015) to estimate the mean and covariance of $\widehat{\mathbf{h}} = [\mathbf{h}_m; \mathbf{h}_M]$, with which to construct a joint Gaussian posterior approximation for $\widehat{\mathbf{h}}$ by moment matching. Then, from the projection in (1), we can derive a Gaussian posterior for $[\mathbf{y}_m; \mathbf{y}_M]$. By further applying Weinstein-Aronszajn identify (Kato, 2013), we can compute the entropy terms in (2) conveniently and efficiently.

Each time, DMFAL maximizes the acquisition function (2) to identify a pair of input and fidelity at which to query. The new example is added into $\mathcal{D}$, and the deep multi-fidelity model is retrained and updated. The process repeats until the maximum number of training examples have been queried or other stopping criteria are met.

3 Batch Multi-Fidelity Active Learning with Budget Constraints

While effective, DMFAL can only identify and query at one input and fidelity each time, hence ignoring the correlations between the successive queries. As a result, highly correlated examples are easier to be acquired and incorporated into the training set. Consider we have found $(\mathbf{x}^*, m^*)$ that maximizes the acquisition function (2). It is often the case that a single example will not cause a significant change of the model posterior $\{q(\mathbf{W}_m)\}$ (especially when the current dataset $\mathcal{D}$ is not very small). When we optimize the acquisition function again, we are likely to obtain a new pair $(\hat{\mathbf{x}}^*, \hat{m}^*)$ that is close to $(\mathbf{x}^*, m^*)$ (*e.g.*, $\hat{m}^* = m^*$ and $\hat{\mathbf{x}}^*$ close to $\mathbf{x}^*$). Accordingly, the queried example will be highly correlated to the previous one. The redundant information within these examples can restrict the learning efficiency, and demand for more queries (and cost) to achieve the satisfactory performance. Note that the similar issue have been raised in other single-fidelity, pool-based active learning works, *e.g.*, (Geifman and El-Yaniv, 2017; Kirsch et al., 2019); see Sec 4.

To overcome this problem, we propose BMFAL-BC, a batch multi-fidelity active learning approach to reduce the correlations and to promote the diversity of training examples, so that we can improve the learning performance, lower the sample complexity, and better save the cost. In addition, we take into account the budget constraint in querying the batch examples, which is common in practice (like cloud or high-performance computing) (Mendes et al., 2020). Under the given budget, we aim to find a batch of multi-fidelity queries that improves the benefit-cost ratio as much as possible.

3.1 Batch Acquisition Function

Specifically, we first consider an acquisition function that allows us to jointly optimize a set of inputs and fidelities. While it seems natural to consider how to extend (2) to the batch case, the acquisition function in (2) is about the mutual information between $\mathbf{y}_m(\mathbf{x})$ and $\mathbf{y}_M(\mathbf{x})$. That means, it only measures the utility of the query $(m, \mathbf{x})$ in improving the estimate of the target function at $\mathbf{x}$ itself (*i.e.*, $\mathbf{y}_M(\mathbf{x})$), rather than at any other input. To take into account the utility in improving the function estimation at all the inputs, we therefore propose a new single acquisition function,

$$a_s(m, \mathbf{x}) = \mathbb{E}_{p(\mathbf{x}')} \left[\mathbb{I} \left(\mathbf{y}_m(\mathbf{x}), \mathbf{y}_M(\mathbf{x}') | \mathcal{D} \right) \right] \tag{3}$$

where $\mathbf{x}' \in \Omega$ and $p(\mathbf{x}')$ is the distribution of the input in Ω. We can see that, by varying the input $\mathbf{x}'$ in the second argument of the mutual information, we are able to evaluate the utility of the query in improving the estimation of the entire body of the target function. Hence, it is more reasonable and comprehensive. Now, consider querying a batch of examples under the budget B, we extend (3) to

$$a_{\text{batch}}(\mathcal{M}, \mathcal{X}) = \mathbb{E}_{p(\mathbf{x}')} \left[\mathbb{I} \left(\{\mathbf{y}_{m_j}(\mathbf{x}_j)\}_{j=1}^n, \mathbf{y}_M(\mathbf{x}') | \mathcal{D} \right) \right], \quad \text{s.t. } \sum_{j=1}^n \lambda_{m_j} \leq B, \tag{4}$$

where $\mathcal{M} = \{m_1, \ldots, m_n\}$, $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, and $\mathcal{D}$ is the current training dataset. We can see that the more correlated $\{\mathbf{y}_{m_j}(\mathbf{x}_j)\}_{j=1}^n$, the smaller the mutual information, and hence the smaller the expectation in (4). Therefore, the batch acquisition function implicitly penalizes strongly correlated queries and favors diversity.

The expected mutual information in (4) is usually analytically intractable. However, we can efficiently compute it with an Monte-Carlo approximation. That is, we draw A independent samples from the input distribution, $\mathbf{x}'_1, \ldots, \mathbf{x}'_A \sim p(\mathbf{x}')$, and compute

$$\widehat{a}_{\text{batch}}(\mathcal{M}, \mathcal{X}) = \frac{1}{A} \sum_{l=1}^{A} \mathbb{I} \left(\{\mathbf{y}_{m_j}(\mathbf{x}_j)\}_{j=1}^n, \mathbf{y}_M(\mathbf{x}'_l) | \mathcal{D} \right). \tag{5}$$

It is straightforward to extend the multi-variate delta method used in DMFAL to calculate the mutual information in (5). We can then maximize (5) subject to the budget constraint, $\sum_{j=1}^B \lambda_{m_j} \leq B$, to acquire more diverse and informative training examples. In addition, when parallel computing resources (*e.g.*, multi-core CPUs/GPUs and computer clusters) are available, we can acquire these queries in parallel to further speed up the active learning.

However, directly maximizing (5) will incur a combinatorial search over multiple fidelities in $\mathcal{M}$, and hence is very expensive. Note that the number of examples n is not fixed, and can vary as long as the cost does not exceed the budget B, which makes the optimization even more challenging. To address these issues, we propose a weighted greedy algorithm to sequentially determine each $(m_j, \mathbf{x}_j)$ pair.

3.2 Weighted Greedy Optimization

Specifically, at each step, we maximize the mutual information increment on one pair of input and fidelity, weighted by the corresponding cost. Suppose at step k, we have identified a set of k inputs and fidelities at which to query, $\mathcal{Q}_k = \{(\mathbf{x}_j, m_j)\}_{j=1}^k$. To identify a new pair of input and fidelity at step $k+1$, we maximize a weighted incremental version of (5),

$$\begin{aligned} \hat{a}_{k+1}(\mathbf{x}, m) = \frac{1}{A} \sum_{l=1}^{A} & \frac{\mathbb{I} \left(\mathcal{Y}_k \cup \{\mathbf{y}_m(\mathbf{x})\}, \mathbf{y}_M(\mathbf{x}'_l) | \mathcal{D} \right) - \mathbb{I} \left(\mathcal{Y}_k, \mathbf{y}_M(\mathbf{x}'_l) | \mathcal{D} \right)}{\lambda_m} \\ \text{s.t. } & \lambda_m + \sum_{\widehat{m} \in \mathcal{M}_k} \lambda_{\widehat{m}} \leq B, \end{aligned} \tag{6}$$

where $\mathcal{Y}_k = \{\mathbf{y}_{m_j}(\mathbf{x}_j) | (\mathbf{x}_j, m_j) \in \mathcal{Q}_k\}$, and $\mathcal{M}_k$ are all the fidelities in $\mathcal{Q}_k$. At the beginning ($k = 0$), we have $\mathcal{Q}_k = \emptyset$, $\mathcal{Y}_k = \emptyset$ and $\mathcal{M}_k = \emptyset$. Each step, we look at each valid fidelity, *i.e.*, $1 \leq \lambda_m \leq B - \sum_{\widehat{m} \in \mathcal{M}_k} \lambda_{\widehat{m}}$, and find the optimal input. We then add the pair $(m, \mathbf{x})$ that gives the largest $\hat{a}_{k+1}$ into $\mathcal{Q}_k$, and proceed to the next step. Our greedy approach is summarized in Algorithm 1.

Algorithm 1 Weighted-Greedy($\{\lambda_m\}$, budget B)

$k \leftarrow 0, \mathcal{Q}_k \leftarrow \emptyset, C_k \leftarrow 0$
while $C_k \leq B$ **do**
  Optimize the weighted incremental acquisition function in (6):

$$(\mathbf{x}^*, m^*) = \underset{\mathbf{x}\in\Omega, 1\leq m\leq B-C_k}{\operatorname{argmax}} \hat{a}_{k+1}(\mathbf{x}, m)$$

  if Infeasible **then**
    break
  end if
  $k \leftarrow k + 1$
  $\mathcal{Q}_k \leftarrow \mathcal{Q}_{k-1} \cup \{(\mathbf{x}^*, m^*)\}$
  $C_k \leftarrow C_{k-1} + \lambda_{m^*}$
end while
Return $\mathcal{Q}_k$

The intuition of our approach is as follows. Mutual information is a classic submodular function (Krause and Guestrin, 2005), and hence if there were no budget constraints or weights, greedily choosing the input which increases the mutual information most achieves a solution within $(1 - 1/e)$ of the optimal (Krause and Golovin, 2014). However, Leskovec et al. (2007) observed that when items have a weight (corresponding to the cost for different fidelities in our case) and there is a budget constraint, then the approximation factor can be arbitrarily bad. We observe, *and prove*, that this only occurs as the budget is about to be filled, and up until that point, the weighted greedy optimization achieves the best possible $(1 - 1/e)$-approximation of the optimal. We can formalize this *near* $(1-1/e)$*-approximation* in two ways, proven in the Appendix. Let OPT(B) be the maximum amount of mutual information that can be achieved with a budget B.

Theorem 3.1. *At any step of Weighted-Greedy (Algorithm 1) before any choice of fidelity would exceed the budget, and the total budget used to that point is $B' < B$, then the mutual information of the current solution is within $(1 - 1/e)$ of OPT(B').*

Corollary 3.1. *If Weighted-Greedy (Algorithm 1) is run until input-fidelity pair $(\mathbf{x}, m)$ that corresponds with the maximal acquisition function $\hat{a}_{k+1}(\mathbf{x}, m)$ would exceed the budget, it selects that input-fidelity pair anyways (the solution exceeds the budget B) and then terminates, the solution obtained is within $(1 - 1/e)$ of OPT(B).*

We next sketch the main idea of how to prove the main result. Adding a new fidelity-input pair $(m, \mathbf{x})$ gives an increment of learning benefit $\Delta_j = \mathbb{I}(\mathcal{Y}_k \cup \{\mathbf{y}_m(\mathbf{x})\}, \mathbf{y}_M(\mathbf{x}'_l)|\mathcal{D}) - \mathbb{I}(\mathcal{Y}_k, \mathbf{y}_M(\mathbf{x}'_l)|\mathcal{D})$. Since we need to trade off to the cost λ_m, we can view there are λ_m independent copies of $\mathbf{x}$ (for the particular fidelity m), and adding each copy gives $\frac{\Delta_j}{\lambda_m}$ benefit. We choose the optimal $\mathbf{x}$ and m that maximizes the benefit $\frac{\Delta_j}{\lambda_m}$. Since all the λ_m copies of $\mathbf{x}$ have the equal, biggest benefit (among all possible choices of inputs in Ω and fidelities in $\mathcal{M}$), we choose to add them first (greedy strategy), which in total gives Δ_j benefit – their benefit does not diminish as each copy is added. Via dividing by λ_m and considering the copies, which each have unit weight, we can apply the standard submodular optimization analysis obtaining $(1 - 1/e)$OPT, at least until we encounter the budget constraint.

3.3 Algorithm Complexity

The time complexity of our weighted greedy optimization is $\mathcal{O}(\frac{B}{\lambda_1}MG)$ where λ_1 is the cost for the lowest fidelity, G is the cost of the underlying numerical optimization (*e.g.*, L-BFGS) and acquisition function computation (detailed in (Li et al., 2022)). The space complexity is $\mathcal{O}(\frac{B}{\lambda_1}(r + d + 1))$, which is to store at most B/λ_1 pairs of input locations and fidelities, and their corresponding outputs (*i.e.*, the querying results). Therefore, both the time and space complexities are linear to the budget B.

4 Related Work

As an important branch of machine learning, active learning has been studied for a long time (Balcan et al., 2007; Settles, 2009; Balcan et al., 2009; Dasgupta, 2011; Hanneke et al., 2014). While many prior works develop active learning algorithms for kernel based models, *e.g.*, (Schohn and Cohn, 2000;

Tong and Koller, 2001; Joshi et al., 2009; Krause et al., 2008; Li and Guo, 2013; Huang et al., 2010), recent research focus has transited to deep neural networks. Gal et al. (2017) used Monte-Carlo (MC) Dropout (Gal and Ghahramani, 2016) to generate posterior samples of the neural network output, based on which a variety of acquisition functions, such as predictive entropy and Bayesian Active Learning by Disagreement (BALD) (Houlsby et al., 2011), can be calculated and optimized to query new examples in active learning. This approach has been proven successful in image classification. Kirsch et al. (2019) developed BatchBALD, a greedy approach that incrementally selects a set of unlabeled images under the BALD principle and issues batch queries to improve the active learning efficiency. They show that the batch acquisition function based on BALD is submodular, and hence the greedy approach achieves a $1-1/e$ approximation. Other works along this line includes (Geifman and El-Yaniv, 2017; Sener and Savarese, 2018) based on core-set search, (Gissin and Shalev-Shwartz, 2019; Ducoffe and Precioso, 2018) based on adversarial networks or samples, (Ash et al., 2019) based on the uncertainty evaluated in the gradient magnitude, *etc*.

Different from most methods, DMFAL (Li et al., 2022) conducts optimization-based, rather than pool-based active learning. That is, they optimize the acquisition function in the entire domain rather than rank a set of pre-collected examples to label. It is also related to Bayesian experimental design (Kleinegesse and Gutmann, 2020). In addition, DMFAL for the first time automatically queries examples at different fidelities, and integrates these examples in a deep multi-fidelity model to improve the active learning efficiency while reducing the cost — which is critical in physical simulation and engineering design. The pioneer work of Settles et al. (2008) empirically studies how the human labeling cost can vary in practical active learning, but does not provide a scheme to identify multi-fidelity queries. Our work is an extension of (Li et al., 2022) to generate a batch of multi-fidelity queries so as to reduce the query correlations, improve the diversity and quality of the training examples, while respecting a given budget constraint. A counterpart in the Bayesian optimization domain is the recent work BMBO-DARN (Li et al., 2021), which considers batch multi-fidelity queries for optimizing a black-box function. From the high-level view, BMBO-DARN and our work employ a similar interleaving procedure, *i.e.*, determining a new batch of queries by optimizing an acquisition function, issuing the queries to collect new examples, and updating the surrogate model. However, both the learning setting and acquisition function are different. More important, we consider the budget constraint while BMBO-DARN does not. Thereby, the computation and optimization techniques of the two works are totally different. The BMBO-DARN uses Hamiltonian Monte-Carlo samples of the single function output prediction and constructs empirical covariance matrices to compute the acquisition function, while our method and (Li et al., 2022) use the multi-variate delta method and matrix identities to overcome the challenge of the high-dimensional outputs. BMBO-DARN uses alternating optimization to search over multiple fidelities, while our work develops a weighted greedy algorithm with additional theoretical guarantees to respect the budget constraint.

5 Experiment

5.1 Solving Partial Differential Equations

We first evaluated BMFAL-BC in several benchmark tasks of computational physics, *i.e.*, predicting the solution fields of partial differential equations (PDEs), including *Heat*, *Poisson*'s, and *Burgers*' equations (Olsen-Kettle, 2011). A numerical solver was used to collect the training examples. High-fidelity examples were generated by running the solver with dense meshes, while low-fidelity examples by coarse meshes. The dimension of the output is the number of the grid points. For example, a 50×50 mesh means the output dimension is $2,500$. We provided two-fidelity queries for each PDE, corresponding to 16×16 and 32×32 meshes. We also tested three-fidelity queries for Poisson's equation, with a 64×64 mesh to generate examples at the third fidelity. We denote the three-fidelity setting by Poisson-3. The input comprises of the PDE parameters and/or boundary/initial condition parameters. The details are provided in (Li et al., 2022). We ran the solver at each fidelity for many times, and computed the average running time. We normalized the average running time to obtain the querying cost at each fidelity, $\lambda_1 = 1$, $\lambda_2 = 3$ and $\lambda_3 = 10$. To collect the initial training dataset for active learning, we generated 10 fidelity-1 examples and 2 fidelity-2 examples for in the two-fidelity setting, and 10, 5, and 2 examples of fidelity-1, 2, 3, respectively, for the three-fidelity setting. The training inputs of the initial dataset were uniformly sampled from the domain. To evaluate the prediction accuracy, for each PDE, we randomly sampled 500 inputs, calculated the ground-truth outputs from a much denser mesh: 128×128 for Burger's and Poisson's and 100×100

for Heat equation. The predictions at the highest fidelity were then interpolated to these denser meshes (Zienkiewicz et al., 1977) to evaluate the error.

Competing Methods. Since currently there is not any budget-aware, batch multi-fidelity active learning approach (to our knowledge), for comparison, we first made a simple extension of the state-of-the-art multi-fidelity active learning method, DMFAL (Li et al., 2022). Specifically, to obtain a batch of queries, we ran DMFAL as it is, namely, each step acquiring one example by maximizing (2) and then retraining the model, until the budget is exhausted or no new queries can be issued (otherwise the budget will be exceeded). We denote this method by (i) `DMFAL-BC`. Note that it is still sequentially querying and training inside each batch, but respects the budget. Next, to confirm the effectiveness of the proposed new acquisition function (3) (based on which, we propose our batch acquisition function (4)), we ran active learning in the same away as `DMFAL-BC`, except the acquisition function is replaced by $\frac{a_s(m,\mathbf{x})}{\lambda_m}$ where a_s is defined by (3). To compute a_s, we used an Monte-Carlo approximation similar to (5), where the number of samples A is 20. We denote this method by (ii) `MFAL-BC`. In addition, we compared with (iii) `DMFAL-BC-RF`, which follows the running of `DMFAL-BC`, but each time, it randomly selects a fidelity m, then maximize the mutual information $\mathbb{I}\big(\mathbf{y}_m(\mathbf{x}), \mathbf{y}_M(\mathbf{x})|\mathcal{D}\big)$ — the numerator of (2) — to identify the corresponding input. Similarly, we tested (iv) `MFAL-BC-RF`, which follows the execution of `MFAL-BC`, but each time, it randomly selects a fidelity m and maximizes $a_s(m, \mathbf{x})$. Again a_s is computed by the Monte-Carlo approximation. For all these methods, we used L-BFGS for the input optimization. Finally, we tested (v) `Batch-FR-BC`, which randomly samples both the fidelity and input at each step (*i.e.*, fully random), until the budget is used up or no more fidelities are available. Then the batch of the acquired examples were added into the dataset altogether to retrain the model. Note that there are other possible acquisition functions, *e.g.*, the popular predictive variances and BALD (Houlsby et al., 2011). Their multi-fidelity versions have been tested and compared against DMFAL in (Li et al., 2022), and turns out to be inferior. Hence, we did not test their possible variants/extensions in our paper.

Settings and Results. All the methods were implemented by Pytorch (Paszke et al., 2019). We followed the same setting as in Li et al. (2022) to train the deep multi-fidelity model (see Sec. 2.2), which employed a two-layer NN at each fidelity, `tanh` activation, and the layer width was selected from $\{20, 40, 60, 80, 100\}$ from the initial training data. The dimension of the latent output was 20. The learning rate was tuned from $\{10^{-4}, 5 \times 10^{-4}, 10^{-3}, 5 \times 10^{-3}, 10^{-2}\}$. We set the budget for acquiring each batch to 20 (normalized seconds), and ran each method to acquire 25 batches of training examples. We tracked the running performance of each method in terms of the normalized root-mean-square-error (nRMSE). We repeated the experiment for five times, and report the average nRMSE *vs.* the accumulated cost (*i.e.*, the summation of the corresponding λ's in each acquired example) in Fig. 2. The shaded region exhibits the standard deviation. As we can see, BMFAL-BC consistently outperforms all the competing methods by a large margin. At very early stages, all the methods exhibit similar prediction accuracy. This is reasonable, because they started with the same training set. However, along with more batches of queries, BMFAL-BC shows better performance, and its discrepancy with the baselines is in general growing. In addition, MFAL-BC constantly outperforms DMFAL-BC. Since they conduct the same one-by-one querying and training strategy, the improvement MFAL-BC reflects the advantage of our new single acquisition function (3) over the one used in DMFAL. Together these results have demonstrated the advantage of our method.

5.2 Topology Structure Optimization

Next, we applied our approach in topology structure optimization, which is critical in many manufacturing and engineering design problems, such as 3D printing, bridge construction, and aircraft engine production. A topology structure is used to describe how to allocate the material, *e.g.*, metal and concrete, in a designated spatial domain. Given the environmental input, *e.g.*, a pulling force or pressure, we want to identify the structure with the optimal property of interest, *e.g.*, maximum stiffness. Traditionally, we convert it to a constraint optimization problem, for which we minimize a compliance objective with a total volume constraint (Sigmund, 1997). Since the optimization often needs to repeatedly run numerical solvers to solve relevant PDEs, the computation is very expensive. We aim to learn a surrogate model with active learning, which can predict the optimal structure outright given the environmental input.

Specifically, our task is to design a linear elastic structure in an L-shape domain discretized in $[0, 1] \times [0, 1]$. The environmental input is a load at the bottom half right and described by two

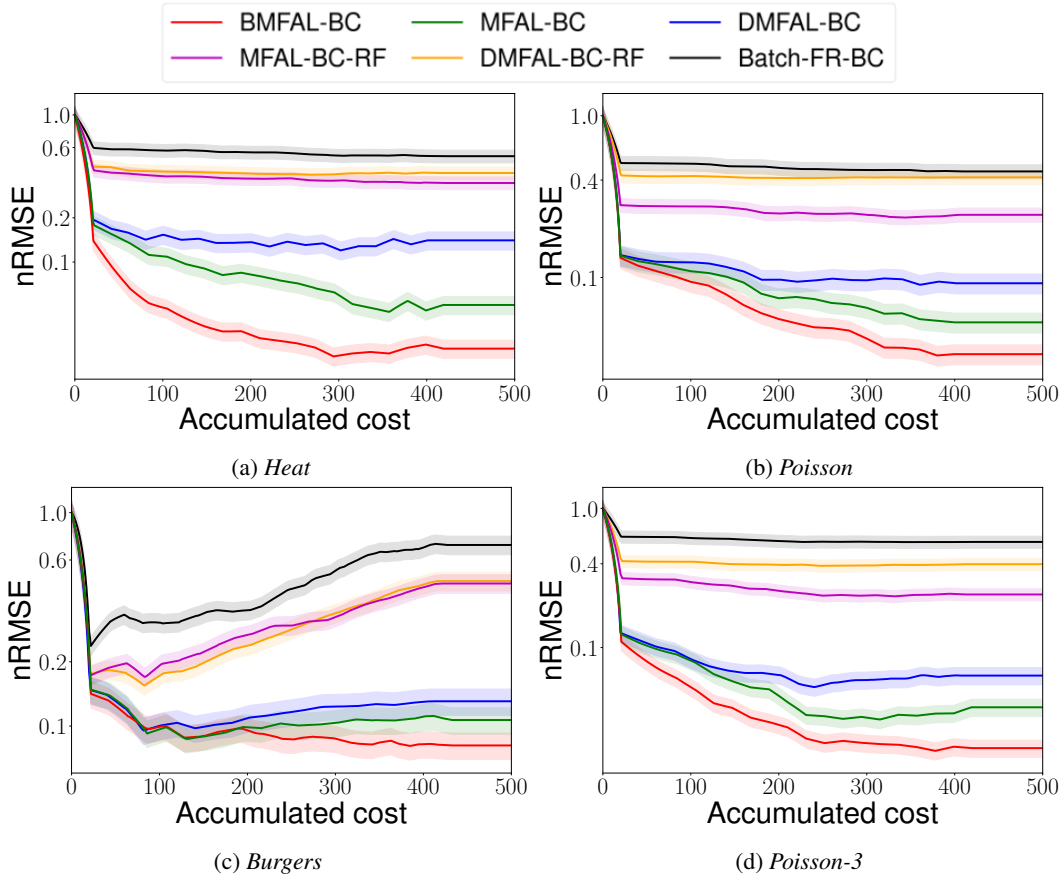


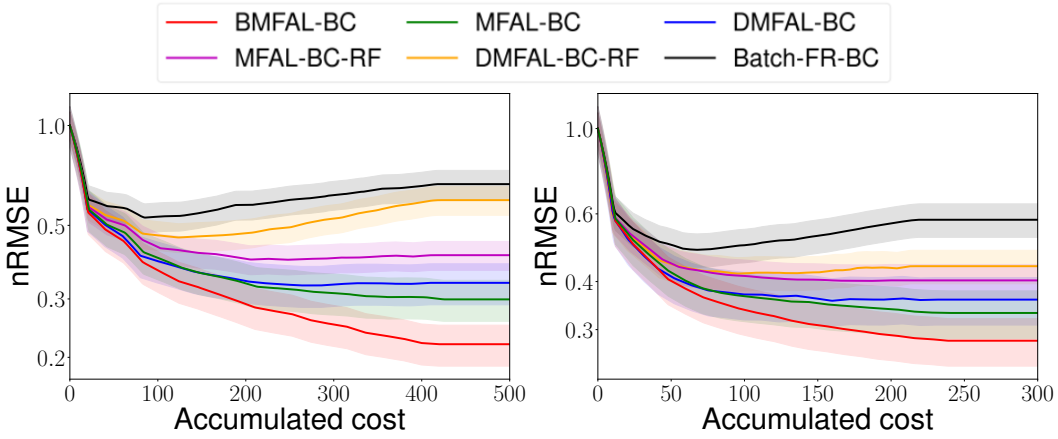
Figure 2: Normalized root-mean-square error (nRMSE) *vs.* accumulated data acquiring cost during batch multi-fidelity active learning, with budget 20 (normalized seconds) per batch. There are two fidelities in acquiring the examples in (a-c) and three fidelities in (d). The results were averaged over 5 runs. The shaded region shows the standard deviation.

parameters: angle (in $[0, \frac{\pi}{2}]$) and location (in $[0.5, 1]$). Given a particular load, we aim to find the structure that has the maximum stiffness. To calculate the compliance in the structure optimization, we need to repeatedly apply the Finite Element Method (FEM) (Keshavarzzadeh et al., 2018), where the fidelity is determined by the mesh. In the active learning, the training examples can be acquired at two fidelities. One corresponds to a 50×50 mesh used in the FEM, and the other 75×75. The output dimension of the two fidelities is therefore $2,500$ and $5,625$, respectively. The querying cost is the normalized average time to find the optimal structure (with the standard approach), $\lambda_1 = 1$ and $\lambda_2 = 3$. To evaluate the performance, 500 test structures were randomly generated with a 100×100 mesh. We interpolated the high-fidelity prediction of each method to the 100×100 mesh and then evaluated the prediction error.

At the beginning, we uniformly sampled the input (*i.e.*, load angle and location) to collect 10 structures at the first fidelity and 2 at the second fidelity, as the initial training set. We then ran all the active learning methods, with budget 20 per batch, to acquire 25 batches of examples. We examined the average nRMSE along with the accumulated cost of acquiring the examples. The results are reported in Fig. 3a. We can see that the prediction accuracy of BMFAL-BC is consistently better than all the competing methods during the active learning. The improvement becomes larger when more examples are acquired. The results confirm the advantage of BMFAL-BC.

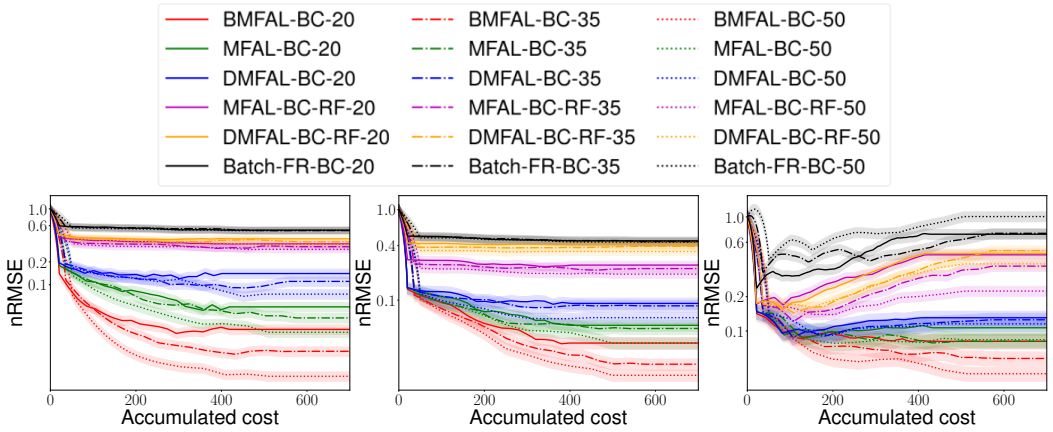
5.3 Predicting Spatial-Temporal Fields of Flows

Third, we evaluated BMFAL-BC in computational fluid dynamics (CFD). We considered a classical application where the flow is inside a rectangular domain (represented by $[0, 1] \times [0, 1]$), and driven



(a) *Topology optimization* (b) *Fluid dynamics*

Figure 3: nRMSE *vs.* the accumulated cost in learning to predict topological structures and fulid dynamics. The budget was set to 20 and 10 for (a) and (b), respectively.



(a) *Heat* (b) *Poisson* (c) *Burgers*

Figure 4: nRMSE *vs.* the accumulated cost under different budgets per batch: $B \in \{20, 35, 50\}$.

by rotating boundaries with a constant velocity (Bozeman and Dalton, 1973). The velocities of different parts inside the flow will vary differently, and eventually lead to turbulent fluids. To compute the velocity field in the time spatial domain, we need to solve the incompressible Navier-Stokes equations (Chorin, 1968), which is known to be computationally challenging, especially under large Reynolds numbers. We considered the active learning task of predicting the first component of the velocity field at 20 evenly spaced points in the temporal domain $[0, 10]$. The training examples can be queried at two fidelities. The first fidelity uses a 50×50 mesh in the $[0, 1] \times [0, 1]$ spatial domain, and the second fidelity 75×75. Accordingly, the output dimensions are $50,000$ and $112,500$; the cost per query is $\lambda_1 = 1$ and $\lambda_2 = 3$. The input is five dimensional, including the velocities of the four boundaries and the Reynold number. The details are given in (Li et al., 2022). For testing, we computed the solution fields of 256 random inputs, using a 128×128 mesh. We used the cubic-spline interpolation to align the prediction made by each method to the 128×128 grid, and then calculated normalized RMSE. To conduct the active learning, we randomly generated 10 and 2 examples in each fidelity as the initial training set. We set the budget to 10, and ran each method to acquire 25 batches. We repeated the experiment for five times, and examined how the average nRMSE varied along with the accumulated cost. As shown in Fig. 3b, BMFAL-BC keeps exhibiting superior predictive performance during the course of active learning. Again, the more examples acquired, the more improvement of BMFAL-BC upon the competing methods. The results are consistent with the previous experiments. Note that throughout these comparisons, we focus on the accumulated

cost of querying (or generating) new examples, because it dominates the total cost, and in practice is the key bottleneck in learning the surrogate model. For example, in topology optimization (Sec 5.2) and the fluid dynamic experiment, running a high-fidelity solver once takes 300-500 seconds on our hardware, while the surrogate training takes less than 2 seconds, and our weighted greedy optimization of the batch acquisition function (Algorithm 1) takes a few seconds. One can imagine for practical larger-scale problems, the simulation cost, *e.g.*, taking hours or even days to generate one example, can be even more dominant and decisive.

5.4 Influence of Different Budgets

Finally, we examined how the budget choice will influence the performance of active learning. To this end, we varied the budget B in $\{20, 35, 50\}$ and tested all the methods for Poisson's, Burger's and heat equations. We used the same two-fidelity settings as in Sec. 5.1. For each budget, we ran the experiment for five times. We show the average nRMSE *vs.* the accumulated cost in Fig. 4. As we can see, the larger the budget per batch, the better the running performance of BMFAL-BC. This is reasonable, because a larger budget allows our method to generate more queries in each batch and in the meantime to account for their correlations or information redundancy. Accordingly, the acquired training examples are overall more diverse and informative. Again, BMFAL-BC outperforms all the competing methods under every budget. The improvement of BMFAL-BC is bigger under larger budget choices. This together demonstrates the advantage of batch active learning that takes into account the correlations between queries.

6 Conclusion

We have presented BMFAL-BC, a budget-aware, batch multi-fidelity active learning algorithm for high-dimensional outputs. Our weighted greedy algorithm can efficiently generate a batch of input-fidelity pairs for querying under the budget constraint, without the need for combinatorially searching over the fidelities, while achieving good approximation guarantees. The results on several typical computational physical applications are encouraging.

Acknowledgments

This work has been supported by MURI AFOSR grant FA9550-20-1-0358 and NSF CAREER Award IIS-2046295. JP thanks NSF CDS&E-1953350, IIS-1816149, CCF-2115677, and Visa Research.

References

Ash, J. T., Zhang, C., Krishnamurthy, A., Langford, J., and Agarwal, A. (2019). Deep batch active learning by diverse, uncertain gradient lower bounds. In International Conference on Learning Representations.

Balcan, M.-F., Beygelzimer, A., and Langford, J. (2009). Agnostic active learning. Journal of Computer and System Sciences, 75(1):78–89.

Balcan, M.-F., Broder, A., and Zhang, T. (2007). Margin based active learning. In International Conference on Computational Learning Theory, pages 35–50. Springer.

Bickel, P. J. and Doksum, K. A. (2015). Mathematical statistics: basic ideas and selected topics, volume I, volume 117. CRC Press.

Bozeman, J. D. and Dalton, C. (1973). Numerical study of viscous flow in a cavity. Journal of Computational Physics, 12(3):348–363.

Chorin, A. J. (1968). Numerical solution of the navier-stokes equations. Mathematics of computation, 22(104):745–762.

Conti, S. and O'Hagan, A. (2010). Bayesian emulation of complex multi-output and dynamic computer models. Journal of statistical planning and inference, 140(3):640–651.

Dasgupta, S. (2011). Two faces of active learning. Theoretical computer science, 412(19):1767–1781.

Ducoffe, M. and Precioso, F. (2018). Adversarial active learning for deep networks: a margin based approach. arXiv preprint arXiv:1802.09841.

Gal, Y. and Ghahramani, Z. (2016). Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In international conference on machine learning, pages 1050–1059.

Gal, Y., Islam, R., and Ghahramani, Z. (2017). Deep bayesian active learning with image data. In International Conference on Machine Learning, pages 1183–1192.

Geifman, Y. and El-Yaniv, R. (2017). Deep active learning over the long tail. arXiv preprint arXiv:1711.00941.

Gissin, D. and Shalev-Shwartz, S. (2019). Discriminative active learning. arXiv preprint arXiv:1907.06347.

Hanneke, S. et al. (2014). Theory of disagreement-based active learning. Foundations and Trends® in Machine Learning, 7(2-3):131–309.

Houlsby, N., Huszár, F., Ghahramani, Z., and Lengyel, M. (2011). Bayesian active learning for classification and preference learning. arXiv preprint arXiv:1112.5745.

Huang, S.-J., Jin, R., and Zhou, Z.-H. (2010). Active learning by querying informative and representative examples. In Advances in neural information processing systems, pages 892–900.

Joshi, A. J., Porikli, F., and Papanikolopoulos, N. (2009). Multi-class active learning for image classification. In 2009 IEEE Conference on Computer Vision and Pattern Recognition, pages 2372–2379. IEEE.

Kato, T. (2013). Perturbation theory for linear operators, volume 132. Springer Science & Business Media.

Kennedy, M. C. and O'Hagan, A. (2000). Predicting the output from a complex computer code when fast approximations are available. Biometrika, 87(1):1–13.

Keshavarzzadeh, V., Kirby, R. M., and Narayan, A. (2018). Parametric topology optimization with multi-resolution finite element models. arXiv preprint arXiv:1808.10367.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114.

Kirsch, A., van Amersfoort, J., and Gal, Y. (2019). BatchBald: Efficient and diverse batch acquisition for deep bayesian active learning. In Advances in Neural Information Processing Systems, pages 7026–7037.

Kleinegesse, S. and Gutmann, M. U. (2020). Bayesian experimental design for implicit models by mutual information neural estimation. In International Conference on Machine Learning, pages 5316–5326. PMLR.

Krause, A. and Golovin, D. (2014). Submodular function maximization. Tractability, 3:71–104.

Krause, A. and Guestrin, C. E. (2005). Near-optimal nonmyopic value of information in graphical models. In Proc. of Uncertainty in Artificial Intelligence (UAI).

Krause, A., Singh, A., and Guestrin, C. (2008). Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. Journal of Machine Learning Research, 9(Feb):235–284.

Leskovec, J., Krause, A., Guestrin, C., Faloutsos, C., VanBriesen, J., and Glance, N. (2007). Cost-effective outbreak detection in networks. In Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 420–429.

Li, S., Kirby, R., and Zhe, S. (2021). Batch multi-fidelity bayesian optimization with deep auto-regressive networks. Advances in Neural Information Processing Systems, 34:25463–25475.

Li, S., Wang, Z., Kirby, R. M., and Zhe, S. (2022). Deep multi-fidelity active learning of high-dimensional outputs. Proceedings of the Twenty-Fifth International Conference on Artificial Intelligence and Statistics.

Li, X. and Guo, Y. (2013). Adaptive active learning for image classification. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 859–866.

Mendes, P., Casimiro, M., Romano, P., and Garlan, D. (2020). Trimtuner: Efficient optimization of machine learning jobs in the cloud via sub-sampling. In 2020 28th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), pages 1–8. IEEE.

Oehlert, G. W. (1992). A note on the delta method. The American Statistician, 46(1):27–29.

Olsen-Kettle, L. (2011). Numerical solution of partial differential equations. Lecture notes at University of Queensland, Australia.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. In Advances in neural information processing systems, pages 8026–8037.

Schohn, G. and Cohn, D. (2000). Less is more: Active learning with support vector machines. In ICML, volume 2, page 6. Citeseer.

Sener, O. and Savarese, S. (2018). Active learning for convolutional neural networks: A core-set approach. In International Conference on Learning Representations.

Settles, B. (2009). Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences.

Settles, B., Craven, M., and Friedland, L. (2008). Active learning with real annotation costs. In Proceedings of the NIPS workshop on cost-sensitive learning, volume 1. Vancouver, CA:.

Sigmund, O. (1997). On the design of compliant mechanisms using topology optimization. Journal of Structural Mechanics, 25(4):493–524.

Tong, S. and Koller, D. (2001). Support vector machine active learning with applications to text classification. Journal of machine learning research, 2(Nov):45–66.

Zienkiewicz, O. C., Taylor, R. L., Zienkiewicz, O. C., and Taylor, R. L. (1977). The finite element method, volume 36. McGraw-hill London.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [Yes]

(b) Did you mention the license of the assets? [N/A]

(c) Did you include any new assets either in the supplemental material or as a URL? [N/A]

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Appendix

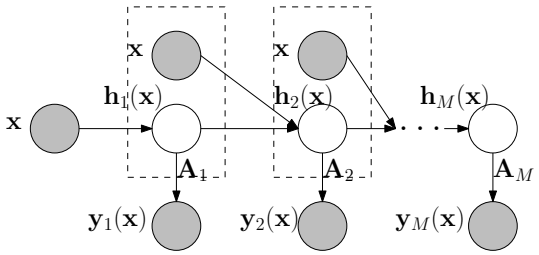


Figure 5: Graphical representation of DMFAL. The low dimensional latent output in each fidelity $\mathbf{h}_m(\mathbf{x})$ $(1 \leq m \leq M)$ is generated by a (deep) neural network.

We now formally prove our main theoretical results on the approximate optimization properties of the Weighted-Greedy algorithm that we have proposed. In particular, these bounds are relative to the optimal algorithm with a budget B, we denote its mutual information as OPT(B). We note that the optimal is with respect to the measurement of $\frac{1}{A}\sum_{l=1}^{A}\mathbb{I}\left(\mathcal{Y}_k, \mathbf{y}_M(\mathbf{x}'_l)|\mathcal{D}\right)$ on the A Monte Carlo samples, and only over the space of inputs Ω and fidelities $\mathcal{M}$ we consider. If more Monte Carlo samples are considered, or somehow mutual information is computed precisely, or more fidelities are searched over, then the OPT(B) considered will increase, and the near-optimality of the greedy algorithm will continue to be approximately proportional to that optimal potential value. Since DMFAL can actively choose an optimal $\mathbf{x} \in \Omega$ for a fixed fidelity m, which is already optimized over a continuous space, the optimal bound we consider OPT(B) is relative to this method.

We now restate and prove the main results.

Theorem A.1 (Theorem 3.1)**.** *At any step of Weighted-Greedy (Algorithm 1) before any choice of fidelity would exceed the budget, and the total budget used to that point is $B' < B$, then the mutual information of the current solution is within $(1 - 1/e)$ of OPT(B').*

Proof. Given a set of elements $\tilde{\Omega}$ and a submodular objective function ϕ, it is well known that if one greedily selects items from $\tilde{\Omega}$ that most increase ϕ at each step, then after t steps, the selected set achieves an objective value in ϕ within a $(1 - 1/e)$-factor of the optimal set of t elements

from $\tilde{\Omega}$ (Krause and Golovin, 2014). Our objective $\frac{1}{A}\sum_{l=1}^{A}\mathbb{I}\left(\mathcal{Y}_k, \mathbf{y}_M(\mathbf{x}'_l)|\mathcal{D}\right)$, where the mutual information is a classic submodular function (Krause and Guestrin, 2005). However, in our setting each item (an input-fidelity pair $(\mathbf{x}, m)$) has a cost λ_m that counts against a total budget B. Our proof will convert this setting back to the classic unweighted setting so we can invoke the standard $(1-1/e)$-result.

Our Weighted-Greedy algorithm instead chooses an $(\mathbf{x}, m) \in \Omega \times \mathcal{M}$ to optimize $\hat{a}_{k+1} = \Delta_{\mathbf{x},m}/\lambda_m$ where $\Delta_{\mathbf{x},m} = \mathbb{I}\left(\mathcal{Y}_k \cup \{\mathbf{y}_m(\mathbf{x})\}, \mathbf{y}_M(\mathbf{x}'_l)|\mathcal{D}\right) - \mathbb{I}\left(\mathcal{Y}_k, \mathbf{y}_M(\mathbf{x}'_l)|\mathcal{D}\right)$ is the increase in mutual information by adding $(\mathbf{x}, m)$. By scaling this $\Delta_{\mathbf{x},m}$ value by $1/\lambda_m$ we can imagine splitting the effect of $(\mathbf{x}, m)$ into λ_m copies of itself, and considering each of these copies as unit-weight elements. We next argue that our Weighted-Greedy algorithm will achieve the same result as if we split each item into λ_m copies, and that the process on these copies aligns with the standard setting.

First lets observe Weighted-Greedy will achieve the same result as if each $(\mathbf{x}, m)$ was split into λ_m copies. When we split each $(\mathbf{x}, m)$ into copies, each maintains the same scaled contribution of $\Delta_{\mathbf{x},m}/\lambda_m$ to our objective function. And we greedily add the item with largest contribution. So if some $(\mathbf{x}, m)$ has the largest contribution $\hat{a}_{k+1}$ in the weighted setting, then so will its unit weight copy in the unweighted setting. In the unit weight setting, after we add the first copy, this may effect the $\Delta_{\mathbf{x}',m'}/\lambda_{m'}$ contribution of some items $(\mathbf{x}', m') \in \mathcal{Q}_k$. By submodularity, all such items have diminishing returns and their contribution cannot increase. However, the unit weight copies of $(\mathbf{x}, m)$ are essentially independent, and so their $\Delta_{\mathbf{x},m}/\lambda_m$ score does not decrease (if we add all λ_m we increase mutual information by a total of $\Delta_{\mathbf{x},m}$). Since no other item can increase its score, and the copies scores do not decrease, if they were selected for having the maximal score, they will continue to have the maximal score until they are exhausted. Hence, if we select one unit weight copy, we will add all of them consecutively, simulating the effect of adding the single weighted $(\mathbf{x}, m)$ at total cost λ_m. Note that by our assumption in the theorem statement, we can always add all of them.

Finally, we need to argue that this unit-weight setting can invoke the submodular optimization approximation result. For integer λ_m and B values, then this unit-weight version runs a submodular optimization with $B' < B$ steps. The acquisition function used to determine the greedy step is $\hat{a}_{k+1} = \Delta_{\mathbf{x},m}/\lambda_m$, but since we have divided each item $(\mathbf{x}, m)$ into unit weight components with independent contribution to the mutual information $\frac{1}{A}\sum_{l=1}^{A}\mathbb{I}\left(\mathcal{Y}_k, \mathbf{y}_M(\mathbf{x}'_l)|\mathcal{D}\right)$ they satisfy submodularity. Then the weight is the same among all items so it can be ignored, and it maps to the standard submodular optimization with B' steps, and achieves within $(1-1/e)$ of OPT(B') as desired. □

Corollary A.1 (Corollary 3.1)**.** *If Weighted-Greedy (Algorithm 1) is run until input-fidelity pair $(\mathbf{x}, m)$ that corresponds with the maximal acquisition function $\hat{a}_{k+1}(\mathbf{x}, m)$ would exceed the budget, it selects that input-fidelity pair anyways (the solution exceeds the budget B) and then terminates, the solution obtained is within $(1-1/e)$ of OPT(B).*

Proof. Consider that the extended Weighted-Greedy algorithm terminates using total $B^+ \geq B$ total budget. By Theorem 3.1, if we had B^+ budget, then this would achieve within $(1-1/e)$ of OPT(B^+). And since OPT(B^+) $\geq$ OPT(B), then this is within $(1-1/e)$ of OPT(B) as well. □

These results imply that the Weight-Greedy algorithm achieves the desired $(1-1/e)$-approximation until we are near the budget, or we slightly exceed it. If the maximal weight item λ_M is close to the full budget, then we are always in this unbounded case – or may need to greatly exceed the budget to obtain a guarantee. However, on the other hand, if λ_M is fixed and the budget B increases, then our bounds become more accurate. In either case we can obtain a score within $(1-\frac{\lambda_M}{B})(1-1/e)$ of the OPT at a budget B – by excluding the part where the greedy choice may exceed the budget. So as λ_M/B goes to 0, then the approximation goes to $(1-1/e)$.

While we have proven these results in the context of the specific approximated mutual information and parameter space $\Omega \times \mathcal{M}$ these nearly $(1-1/e)$-optimal results will apply to any submodular optimization function, scaled by its optomal cost with a budget B.

Note that Leskovec et al. (2007) proposed another approach to dealing with this budgeted submodular optimization. They proposed to run two optimization schemes, one the method we analyze, and one that simply chooses the items that maximize $\Delta_{\mathbf{x},m}$ at each step while ignoring the difference in their cost λ_m. They show that while the first one may not achieve $(1-1/e)$-approximation, one of these

schemes must achieve that optimality. The cost of running both of them, however, is twice the budget, so in the worst case this combined scheme only achieves within $(1/2)(1 - 1/e)$ of the optimal. This run-twice approach is also wasteful in practice, so we focused on showing what could be proven (near $(1 - 1/e)$-approximation) of just Weighted-Greedy. In fact, as long as $\lambda_M/B \leq 1/2$, we already match their worst case bound.