FINE-TUNING FROM LIMITED FEEDBACKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Instead of learning from scratch, fine-tuning a pre-trained model to fit a related target dataset of interest or downstream tasks has been a handy trick to achieve the desired performance. However, according to the study of Song et al. (2017), standard fine-tuning may expose the information about target data if the pre-trained model is supplied by a malicious provider. Instead of reckoning that data holders are always expert to select reliable models and execute fine-tuning themselves, this paper confronts this problem by exploring a new learning paradigm named Fine-Tuning from limited FeedBacks (FTFB). The appealing trait of FTFB is that the model tuning does not require directly seeing the target data but leveraging the model performances as feedbacks instead. To learn from query-feedback, we propose to fine-tune the pre-trained model on the parameter distribution with the gradient descent scheme. For the deep models whose tuning parameters distribute across multiple layers, a more query-efficient algorithm is further designed which refines the model layer by layer sequentially with importance weight. Extensive experiments on various tasks demonstrate that the proposed algorithms significantly improve the pre-trained model with limited feedbacks only. For downstream tasks which adopt inconsistent evaluation measurement with pre-training, such as fairness or fault-intolerance, we verify our algorithms can also reach good performance.

1 INTRODUCTION

Fine-tuning has emerged as a powerful and widely used technique to learning tasks with insufficient labelled data (Donahue et al., 2014; Devlin et al., 2019; Chen et al., 2020b; Shachaf et al., 2021). Typically, a model pre-trained on a source task can be further used to tune on related downstream tasks where relatively fewer samples are available. Thus, compared with training from scratch, this simple practice does not need huge efforts but still produces satisfactory generalization models.

In this paper, we emphasize that although numerous pre-trained machine learning models are available in recent years, they are not always trusted to deploy on target data for fine-tuning. One important observation is that an intentionally back-door model can extract much data via various secret intrigues if the model is assumed to be accessible after fine-tuning (Song et al., 2017). For example, target data could be encoded in the least significant (lower) bits of the deep model parameters (white-box) or the label vector of augmented data (black-box). In such cases, data holders who are not machine learning experts but care about data privacy during tuning are easily deceived.

In practice, non-expert data holders may not adjust model parameters themselves, which comes to a more general scenario as follows. Alice is a data holder who wants to obtain a predictive model for her sensitive dataset. Bob possesses a model established based on a collection of public data. Since Alice is neither able to develop a model herself nor inclined to share the data, she would like Bob to adapt his model to her dataset under some mutually agreed protocols. For instance, Bob is an ML service provider and Alice is a client. Even Bob is a potential attacker who wants to extract Alice's data, he still manages to tune his model to meet Alice's need as he gets paid. We note that this general two-party scenario absorbs the aforementioned special case where data holders execute tuning process themselves given a suspicious pre-trained model.

One can see that the above data extraction happening in fine-tuning is related to the response of the model. In this perspective, although often studied in a one-to-many context such as distributed learning or federated learning (Popov et al., 2018; Jang et al., 2020), barely returning model gradients over target data to do tuning seems to be a workaround. Unfortunately, gradients still carries much

information which suffer data recovering risk (Zhu et al., 2019; Yin et al., 2021) even if there is not an attacker intentionally causing back-doors. But this inspires us to rethink what is a good protocol? The model response should be informative for tuning while not leaking the target data.

Query opens a window for users to visit the preserved information. To enable Bob to fine-tune the model on his side and simultaneously guarantee that Alice's data is least leaked, we propose the setting of Fine-Tuning with limited FeedBacks (FTFB). Specifically, every time Bob submits a candidate model to Alice as a query, Alice runs it on her dataset and returns the according performance to Bob. This query-feedback procedure is not terminated until the satisfactory model performance is achieved or the maximum query number is reached. We notice that some recent research (Turner et al., 2021) tunes hyperparameter in a similar setting, but their goal is to fit both training and validation set and they cannot directly apply to the optimization for thousands of parameters either.

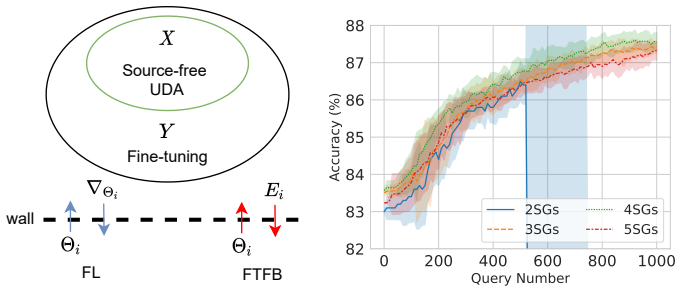
We focus on how to deal with this restricted setting. In FTFB, we aim to leverage the query-feedback information to fine-tune the pre-trained model so that it will gradually approach the desired unknown model on Alice's data. Our main idea is to characterize the geometry of model performance w.r.t. model parameters and utilize the estimated gradient information to search for the better models. For modern deep neural networks whose key parameters distribute on different layers, we introduce the thought of sequentially layerwise tuning and importance weighting, which turns out dramatically saving the query budget.

Our contributions

- Motivated by remedying data extraction problem in fine-tuning, we introduce the setting of Fine-Tuning with limited FeedBacks (FTFB) which leverages limited historical model performances as feedbacks to update pre-trained model. By limiting the length of performance score and query number, we derive that this setting is flexibly controlled for information leakage about target data.
- To produce good generalization models for a tight query budget, we tailor-design the optimization algorithm called Performance-guided Parameter Search (PPS) which fine-tunes the pre-trained model via estimated gradients. Regarding tuning deep neural neural networks, Layerwise Coordinate Parameter Search (LCPS) is further presented towards the query-efficiency goal.
- Besides experimentally justifying the proposed FTFB setting on different datasets including tabular data, text and images, we verify the performance of our algorithms on different target tasks. In particular, on image classification task, our method that only uses limited feedbacks even rival the recent work (Wang et al., 2020) that accesses entire features of target data.

2 Fine-Tuning from Limited FeedBacks (FTFB)

Generalizing a pre-trained model to the target task motivates the model fine-tuning setting. Fig. 1(a) compares FTFB with three most related learning settings from the aspect of *what kind of target data information is accessible for model tuning*. If the target data is labelled, this literally comes to a supervised learning paradigm, i.e., the well-known fine-tuning technique (Donahue et al., 2014). As source data typically does not involve in tuning stage, fine-tuning is viewed related to the setting of source-free Unsupervised Domain Adaptation (UDA) (Sahoo et al., 2020; Li et al., 2020a; Liang et al., 2020; Wang et al., 2020). As all these works do not consider data privacy, target data would be easily extracted by back-doors model even tuning process is not observed by model provider (Song et al., 2017). Federated Learning (FL) (Shokri & Shmatikov, 2015) allows global model to fit local data by asking the derived gradients instead of uploading data externally. This thought is therefore used to fine-tune the general model to user private data (Popov et al., 2018). Although studied in a one-to-many sense, this setting seems as a workaround for our



(a) (b)

Figure 1: (a) Learning settings comparison which differ by what kind of information about target data is accessible. (b) Algorithm sensitivity in terms of the precision of feedbacks

challenge. Unfortunately, data extraction can benefit from high-dimensional gradients, and the clean gradients promote the gradient inversion technique (Yin et al., 2021) as well. Notably, the proposed FTFB neither accesses target data features nor the corresponding labels; it instead only requires limited evaluation performances of candidate models as feedbacks, such as test error or accuracy.

Although model performances are less informative compared with gradients, they can still be used to intentionally encode some secrets about target data. However, we are able to avert this kind information leakage by controlling the number of SGs (Significant Figures) for model evaluation results. To confirm this thought we take classification accuracy as an example, and run our proposed algorithm PPS (introduced in Section 3) by setting the number of SGs for accuracy from 2 to 5. The experimental results are shown as Fig. 1(b) which suggests that 2-SG accuracy tends to collapse during the tuning and 3-SG accuracy (e.g., 84.2%) is sufficient for model tuning. In other words, model providers use 2-SG accuracy for tuning and leave the lowest one SG for secretes encoding will fail in this task.

3 Proposed Method

Now we (Bob's role) figure out how to utilize query-feedback information to fine-tune a given pre-trained model under the setting of FTFB. The pre-trained model is denoted by F_{Θ^0} which is parameterized with Θ^0. The optimization objective of pre-training is any feasible function which could be agnostic to data holder. Let $E(\cdot)$ denote the evaluation function which measures the classification performance on target data. To adapt the pre-trained model Θ^0 to target data $\mathcal{D}$, we need to solve the following problem

$$\Theta^* = \arg\min_{\Theta} E(\mathcal{D}; \Theta) = \arg\min_{\Delta} \frac{1}{|\mathcal{D}|} \sum_{(x_i, y_i) \in \mathcal{D}} \mathbb{I}(F(x_i; \Theta^0 + \Delta) \neq y_i), \tag{1}$$

where $\mathbb{I}(\cdot)$ is the sign function, and Δ represents the difference between the pre-trained model F_{Θ^0} and the potential optimal model F_{Θ^*}. In this sense, Δ is expected to compensate for the actual discrepancy between source and target. If $\mathcal{D}$ is accessible, the standard fine-tuning is adopted to optimize the problem (1) by first differentiating the sign function and then tuning the model with the end-to-end back-propagation. In the proposed FTFB, however, $\mathcal{D}$ is merely visited by queries, making the model evaluation isolated from the tuning process. But we still have chance to approach the optimal model F_{Θ^*} by crafting promising updated models. For example, a simple and direct strategy is to exhaustively vary model parameters and select the best one according to feedbacks. This method is however impractical when we do not have any prior about the target data and thus the query number in this case will be too large to tolerate.

Denoting the query number by q and the computation cost of each forward pass by m, we point out the reasons for only limited feedbacks are three folds. (1) minimize the communication cost $q(|\Theta| + 1)$, (2) reduce the inference computation qm, (3) prevent unexpected data leakage. Note that in the fine-tuning task, model parameters carry sufficient information to be queries, so we do not need to upload the entire model (See Fig. 1(a)). The former two reasons are quite straightforward, which is also referred in distributed learning work (Popov et al., 2018). As we have presented that 3-SG performance values cannot provide much space for data extraction in Fig. 1(b), other types of information leakage are supposed to be small by again limiting the query number. A more concrete example is discussed in Section 6.

According to Nesterov & Spokoiny (2017), the desired query number q is supposed to be positively correlated with the dimension of model parameters $|\Theta|$. In real applications (Sun et al., 2019; Liang et al., 2020; Wang et al., 2020), only a small proportion of model parameters, i.e., $\theta \subseteq \Theta$, are typically tuned. The query number q is thus acceptable especially when $|\theta| \ll |\Theta|$ holds (Wang et al., 2020). For convenience, the tuned parameters θ are dubbed as "key parameters" used in the rest of the paper.

3.1 Performance-guided Parameter Search

In a general FTFB setting, the specific form of evaluation function $E'(\cdot)$ could be agnostic to model provider, but it is still required to know whether the minimized or maximized feedback score is preferred (Refer to the experiments in Section 6). Although problem (1) cannot be optimized with an end-to-end manner, we realise that data holder, i.e., Alice, executes an evaluation function $E'(\cdot)$ in

a similar way to the computation of the loss term in Eq. (1). That means the feedbacks imply the information about both the target data $\mathcal{D}$ and evaluation function $E'(\cdot)$.

Our main idea is continuously exploring the promising direction from a collection of query-feedback pairs to update key parameters θ under the gradient descent scheme. Inspired by black-box optimization (Audet & Hare, 2017), our work focus on tuning model parameters. Concretely, taking advantages of the reparameterization trick in Natural Evolution Strategies (NES) (Wierstra et al., 2014), we propose to treat key parameters θ as a random variable and search for the optimal value by characterizing its distribution. Let ω denote the parameters of density $\pi(\theta|\omega)$. Suppose we are told that the minimized evaluation function $E'(\cdot)$ is preferred. Instead of directly minimizing its value, we minimize its expectation under the search distribution, i.e., density $\pi(\theta|\omega)$,

$$J(\omega) = \mathbb{E}_{\pi(\theta|\omega)}[E'(\mathcal{D};\theta)] = \int E'(\mathcal{D};\theta)\pi(\theta|\omega)\, d\theta. \tag{2}$$

Note θ is a random variable and Eq. (2) is a function w.r.t. ω. Minimizing Eq. (2) derives the gradient:

$$\nabla_\omega J(\omega) = \mathbb{E}_{\pi(\theta|\omega)}[E'(\mathcal{D};\theta)\nabla_\omega \log \pi(\theta|\omega)] \simeq \frac{1}{b}\sum_{i=1}^{b} E'(\mathcal{D};\theta_i)\nabla_\omega \log \pi(\theta_i|\omega) \tag{3}$$

The left-hand equation uses the so called 'log-likelihood trick' which enables the gradient decoupled from the evaluation function $E'(\cdot)$. The right-hand expression is attained by using the sampled $\theta_1, ..., \theta_b$ to estimate the gradient of density parameter ω.

In a manner similar to NES, $\pi(\cdot)$ is instanced by Gaussian distribution for closed-form derivation. By using the reparameterization trick, we sample the query model for the first round and update around the current model θ^0, i.e., $\theta^1 = \theta^0 + \sigma\delta$, where σ is the variance and $\delta \sim \mathcal{N}(0, I)$. To reduce sampling variance, we adopt antithetic sampling (Geweke, 1988) which is demonstrated to stabilize the update. That is, we always sample a pair of Gaussian noises δ_j and $\delta_{b-j+1} = -\delta_j$, generating a population of b perturbations totally. As a result, gradient estimation with these points yields the following stochastic gradient descent update,

$$\theta^{t+1} \leftarrow \theta^t - \frac{\eta}{\sigma b}\sum_{i=1}^{b} \delta_i E'(\mathcal{D};\theta^t + \sigma\delta_i) \tag{4}$$

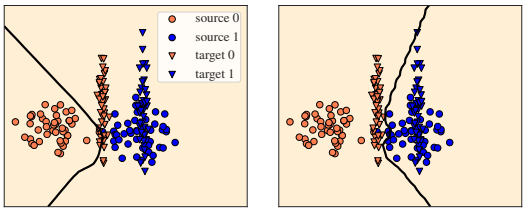
Eq. (4) presents a first-order optimization method in which $E'(\mathcal{D};\theta_t + \sigma\delta_i)$ is a scalar. The high order information could be taken into consideration for obtaining more precise gradient if $|\theta|$ is not large, which is left to our future work. Alg. 1 summarizes this procedure, called Performance-guided Parameters Search (PPS). Particularly, to reduce the impact of the scale of model performance, in practice we normalize the feedbacks before using them, following the strategy of Li et al. (2019).

Algorithm 1 Performance-guided Parameter Search (PPS)

Input: variance σ, query budget q, batch size b

1: **for** $t = 0, ..., \lfloor q/b \rfloor$ **do**
2: Sample $\{\delta_j\}_{j=1}^{b/2} \sim \mathcal{N}(0, I)$, and for each j get $\delta_{b-j+1} = -\delta_j$
3: Generate candidate models $\{\theta_i\}_i^b$ as queries where $\theta_i = \theta^t + \sigma\delta_i$
4: Collect feedbacks $\{E'(\mathcal{D};\theta_i)\}_{i=1}^b$
5: Update θ^{t+1} by Eq. (4)
6: **end for**

Output: $\theta^{\lfloor q/b \rfloor + 1}$



(a) Pre-training (b) FTFB on target

Figure 2: Example of FTFB optimized by PPS.

We present a toy example to verify the algorithm PPS. A 3-MLP network is firstly pre-trained on source data (two Gaussians with the variance of $[0.7, 0.7]$) and is then fine-tuned on target data (another two Gaussians with the variance of $[0.1, 1.5]$) with model test error as feedback. Fig. 8(d) presents that the classifier is able to correctly classify two classes of source data after pre-training, but fails on target data. By empirically setting the last layer network as key parameters, we fine-tune the pre-trained classifier following the steps of Alg. 1, where only a half of randomly selected target data are used to evaluate the query models. Given a query budget as 400, Fig. 8(b) shows that the pre-trained classifier finally adapts to the target data successfully. As most model parameters are frozen during fine-tuning, we observe that the tuned model eventually maintains a good classification performance on source task as well.

3.2 LAYERWISE COORDINATE PARAMETER SEARCH

In many applications, θ often consist of parameters distributed on different layers of deep neural networks. For example, one can narrow the domain discrepancy by adjusting the batch normalization layers (Li et al., 2016; Wang et al., 2020). Without loss of generality, we decompose key parameters layerwise, i.e., $\theta = \{\theta_1, \theta_2, ..., \theta_K\}$, where $\theta_k (1 \leq k \leq K)$ represents the k-th layer parameters. Obviously, treating θ as an entirety to do sampling ignores the intrinsic connections among the chained parameters; parameters on later layers cannot immediately capture the change happening in the former layers in each round gradient estimation. As a result, the model converge slowly and cannot get good performance with given a tight query budget.

Based on the above observation, we tune key parameters θ more naturally, following the thought of greedy search. Specifically, every time we only focus on updating a single layer's parameters θ_k while freezing the remaining parameters, i.e., $\theta - \{\theta_k\}$. This is similar to sequentially training neural networks (Belilovsky et al., 2019). However, they try to scale to accessible large dataset training while we are targeting the query-efficient tuning. Furthermore, recent works (Kirkpatrick et al., 2017; Li et al., 2020b) claim that different layers should have different levels of impact, which inspires us to assign more query budget to important layers. Technically, we define the sampling probability of k-th layer in the form of

$$p_k = \frac{e^{\alpha_k}}{\sum_{i=1}^{K} e^{\alpha_i}}, \tag{5}$$

where α_k denotes the importance of the k-th layer. Instead of applying the static weight, we follow the idea of Exp3 algorithm in multi-arm bandit (Seldin et al., 2013) and propose a new adaptive update rule for the layer importance: by first treating the average improvement in each step as the immediate reward and maximizing the cumulative reward, i.e., total improvement,

$$\alpha_k^{t+1} = \alpha_k^t + \beta(\bar{E'}_k^{t+1}(\mathcal{D}; \theta) - \bar{E'}_{k-1}^{t+1}(\mathcal{D}; \theta)), \tag{6}$$

where $\bar{E'}_k^{t+1}(\mathcal{D}; \theta)$ denotes the average function value over the queries of k-th layer at round $t+1$. It turns out achieving small cumulative regret as claimed by Exp3 algorithm. To obtain the sampling probability, we keep a unit execution for every layer, where u[1] quires are consumed. The complete algorithm, named Layerwise Coordinate Parameter Search (LCPS), is formally summarized into Alg. 2. We clarify three points about this algorithm. (1) For clearer statement, we introduce the auxiliary variable I_k^{t+1} to denote the immediate reward for $(t+1)$-th iteration (Step 4). (2) For each outer iteration, θ_k is at least updated once, denoted by $\theta_k^{t+\frac{1}{2}}$ (Step 3). (3) In practice, we recycle the remaining queries if they are not run out during the reassignment (Step 8-11).

Algorithm 2 Layerwise Coordinate Parameter Search (LCPS)

Input: Pre-trained model θ, query budget q, learning rates η, β, batch size b, variance σ, unit queries u

1: **for** $t = 0, ..., \lfloor q/b \rfloor$ **do**
2: **for** $k = 1, ..., K$ **do**
3: Update $\theta_k^{t+\frac{1}{2}}$ with u queries by Eq. (4)
4: Compute $I_k^{t+1} = \bar{E'}_k^{t+1}(\mathcal{D}; \theta) - \bar{E'}_{k-1}^{t+1}(\mathcal{D}; \theta)$
5: **end for**
6: $\alpha_k^{t+1} \leftarrow \alpha_k^t + \beta I_k^{t+1}$ # Eq. (6)
7: Compute p^t by Eq. (5)
8: **for** $j = 1, ..., \lfloor (b - Ku)/u \rfloor$ **do**
9: Sample a layer k with p_k^t
10: Update θ_k^{t+1} with u queries by Eq. (4)
11: **end for**
12: **end for**

Output: $\theta^{\lfloor q/b \rfloor + 1}$

4 EXPERIMENT

4.1 DATASETS AND EXPERIMENTAL SETUP

The datasets in our experiments are organized as follows. The first two datasets UCI-Adult (Kohavi, 1996) and Review (McAuley & Leskovec, 2013) are used following the work (Chen et al., 2020a) which studies privacy leakage in deep transfer learning. The corrupted CIFAR-10 (Hendrycks & Dietterich, 2019) is employed following the source-free UDA work (Wang et al., 2020).

[1]As the number of parameters in different layers varies, u is not identical for different layers in practice. We use the same u for the convenient statement here.

Adult Census Income dataset comes from the UC Irvine repository which has 14 properties such as country, age, work-class,education, etc. We use the records with country of "U.S" to pre-train a binary classifier and use "non-U.S" records as unobserved target data. Review dataset is constructed from Amazon review data with two categories of products selected, i.e., "Electronics" and "Watches". In our setting, the data-rich category "Electronics" is used to pre-train a prediction model which maps user comments to the rate score ranging from one to five, and "Watches" is treated as the target data. CIFAR-10-C is original developed to benchmark the model robustness and different types of corruption have been included. We use these corrupted images to mimic the inaccessible target data which has an unexpected distribution shift with the clean source data.

Throughout all above datasets, the target data are equally split into two parts. One is *support set* that is used for evaluating the query model during fine-tuning process, and the other is *holdout set* on which the model generalization is assessed.

Implementation Details. Specifically, we apply 3-MLP for UCI-Adult with the penultimate layer of 80 nodes. Following the thought of Kristiadi et al. (2020), we simply perturb the weights of the last layer which exactly contains 80 parameters for binary classification. Regarding the score prediction on Amazon reviews, our implementation is based on the $\mathrm{torchtext}$ library, in which the first layer is a mapping from the vocabulary to a latent representation, followed by three convolutional layers and a linear transformation to label space. The weights of the first layer are treated as the key parameters because they are found to contribute more to the task compared with other layers (This hindsight option is indeed attained by the experimental findings shown as Appendix B.2). For corrupted CIFAR-10 we use residual networks (He et al., 2016) with 26 layers which is implemented by pycls library (Radosavovic et al., 2019). Following Wang et al. (2020), we modulate features on target data by estimating normalization statistics and update transformation parameters channel-wise. It turns out the tuned parameters make up less than 1% of the whole model.

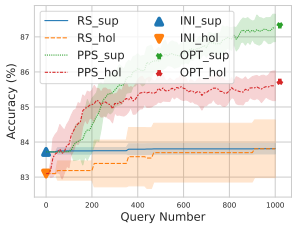
4.2 Setting Justification

We experimentally justify the proposed FTFB setting by conducting our algorithms on different tasks. Besides the naive baseline of directly deploying the pre-trained model on target data (dubbed as 'INI' for 'initial') and standard fine-tuning (dubbed as 'OPT' for 'optimal'), we include Random Search (RS) strategy (Bergstra & Bengio, 2012) as another compared method (refer to Appendix B.5 for details of RS).

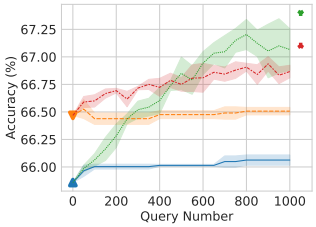
Income classification. We apply Alg. 1 to the Adult dataset with the query budget $q = 1K$. Fig. 3(a) shows the classification performance on support set and holdout set, respectively. We observe that PPS significantly improves the performance of INI and approaches the OPT at $q = 1K$. RS only shows a subtle improvement to pre-trained model and thus not query-efficient in this task.

Rate prediction. Alg. 1 is also run on Amazon where different vocabularies are available. If a good vocabulary is carefully selected (Fig. 3(b)), the improvement space w.r.t. pre-trained model is quite limited. The accuracy implemented by PPS only increases 1.3% and 0.4% on support set and holdout set respectively, even $1K$ queries are consumed. By contrast, if no good vocabulary is applied shown as Fig. 3(c), we observe that PPS rapidly boosts the pre-trained model within a smaller number of queries. Again, PPS is superior to RS by around 5% when $q = 200$. However, PPS converges after 200 queries which has a big gap with OPT's performance. This failure case implies that PPS cannot escape from a local optimum in this task.

Corrupted image classification. We run Alg. 2 on CIFAR-10-C with the query number of $1K$ to tune normalization layers. In this group of experiments, more baselines are included, such as test-time Batch Normalization (BN) (Ioffe & Szegedy, 2015), and test-time adaptation work Tent (Wang et al., 2020). Fig. 4(a) presents the average errors of tuning and testing performance for different methods. Generally, OPT obtains the best performance by using support data for end-to-end tuning. Given $q = 1K$, the average error of LCPS is observed better than Tent which accesses entire features of the support set. Let us have a close look to the specific results over each type of corruption in Fig. 4(b). It is observed that Tent updates towards a wrong direction on some particular corruptions, such as 'motion' and 'bright', making even worse performance compared with BN. However, this performance drop does not happen to ours as LCPS implicitly learns from true label information.



(a) Adult



(b) Amazon (good INI)

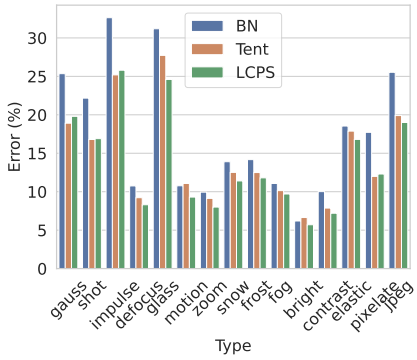
# Query	0	20	40	100
Support	48.0	51.8	54.5	55.8
Holdout	48.5	52.5	55.6	57.2
# Query	200	300	RS	OPT
Support	56.7	56.8	51.5	61.5
Holdout	57.7	58.0	52.7	62.7

(c) Amazon (bad INI)

Figure 3: Performance comparison on (a) Adult and (b-c) Amazon. Note that 'good INI' and 'bad INI' correspond to the different selections of vocabulary.

Method	Support set	Holdout set
INI	40.8	41.1
BN	17.2	17.3
RS ($q = 3K$)	16.7	17.0
PPS ($q = 3K$)	15.1	15.5
LCPS ($q = 1K$)	13.8	13.9
Tent (access X)	14.4	14.8
OPT (access X, Y)	8.5	8.8

(a) Average error (%) over 15 types of corruptions, where RS, PPS, LCPS and Tent are implemented based on test-time BN.



(b) Error rate (%) for each corruption.

Figure 4: (a) Comparison of different model adaptation methods on CIFAR-10-C dataset for the highest severity. (b) Performance of LCPS and Tent on CIFAR-10-C in terms of various corruptions.

Overall, the above experiments justify that our proposed setting of FTFB can be implemented by algorithms PPS and LCPS, which forges a cornerstone for this new fine-tuning setting.

4.3 Beyond Standard Accuracy

The evaluation function $E'(\cdot)$ on target data may be different from that during pre-training. In this subsection, we study downstream tasks which post extra requirements about fairness or fault-intolerance.

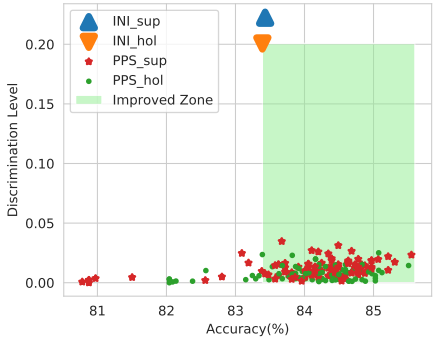


Figure 5: Discrimination level reduction for model fairness tuning.

Fairness. The Adult dataset contains sensitive attributes, e.g., gender and country, and is often used for model fairness (Zafar et al., 2017) assessment. In this experiment, *demographic parity* is adopted as the fairness metric. Suppose data holder aims to obtain a classifier which is expected to be unbiased on gender z ($z = 1$ denotes male and $z = 0$ is for female) for a higher salary ($> \$5k$ annual year), and the corresponding discrimination level of demographic parity is then defined by $\Gamma(\mathcal{D}; \theta) = |P(F_\theta = 1|z = 1) - P(F_\theta = 1|z = 0)|$. In this case, the data holder returns a 2D tuple with one entry for accuracy and the other for discrimination level, i.e., (E', Γ). As two metrics compete with each other, we alternatively update their respective gradient as Alg. 3 in Appendix B.6.

Fig. 5 shows the results of 100 executions of PPS under the above setting. Every star and point denotes the converged model on support set and holdout set respectively. The lower-right is preferred. At the first sight, model performance on support set and holdout set are approximately equally distributed, implying the good model generalization of our method. It is also observed that the pre-trained model reaches the discrimination level of 0.20, which is dramatically reduced to a low level (< 0.05) by PPS. The points falling in the green region refer to the models which achieve considerable improvement

over the pre-trained model in terms of both classification accuracy and model fairness, making up about 90% of the whole set.

Fault-intolerance. For fault-intolerance problems such as medical diagnosis, top-K metric rather than single output prediction is required for producing lower error. We take multi-class classification task over CIFAR-10-C as an example. Suppose that data holder expects a lower classification error on top-5 predictions. In experiment, we replace the standard error with the top-5 error, and report the results on two corruption types, i.e., Gaussian and Impulse noise.

Table 1: Evaluation error (%) with top-1 and top-5 metric for FTFB on two types of corruptions (Gaussian and Impulse noises) over CIFAR-10-C.

Corruptions		Gaussian		Impulse	
		Top-1	Top-5	Top-1	Top-5
Evaluation error	Top-1	20.0 ± 0.13	1.9 ± 0.03	25.9 ± 0.09	3.1 ± 0.05
	Top-5	21.5 ± 0.18	1.7 ± 0.04	30.5 ± 0.14	2.9 ± 0.03

Table 1 exhibits the errors of tuning with top-1 and top-5 evaluation functions respectively. The results show that : 1) Although top-1 evaluation has already achieved small error on top-5 metric, this error could be further reduced by the the proposed LCPS when top-5 is directly used as the evaluation metric under the FTFB setting. 2) However, it is with the sacrifice of other metric. Notably, for impulse noise corruption, the error decreases by 0.2 on top-5 but its top-1 error increase by 4.6%.

The applied fairness metric is a group level measure which is hard to optimize for standard sample-wise loss. Top-K error is non-differentiable which is implemented by extra operation like truncation. That means they both need extra consideration in standard fine-tuning task, but could be fully innocently used in our setting.

4.4 Ablation Study

We empirically verify the three important factors that may have impact on the results, and leave the discussion about other parameters to Appendix B.4.

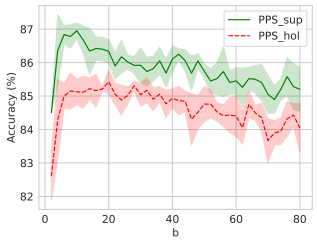
Batch size. We vary the batch size of the Adult dataset from 2 to 80 and collect the accuracy in Fig. 6(a). In terms of the support set, the optimal performance reaches when the batch size is 10. In terms of the holdout set, it achieves its optima with the batch size ranging from 8 to 20, which is consistent with the suggested setting of $4 + \lfloor 3 \log d \rfloor$ (d is the dimension) in Wierstra et al. (2014). It is important to confirm this point as proper batch size saves query budget.

Support set. We explore the effects of the size of support set by varying its ratio from 10% to 90% on Adult target data, and theirs results are shown as Fig. 6(b). As the size of support set increases, it becomes harder to fit all the supported samples given the query budget, but model generalization, i.e., the accuracy on holdset set, improves gradually. Additionally, we can see smaller size of the support set leads to larger variance. In particular, when more than 50% of full support data (> 1000 samples) is used, the model generalization becomes steady with a slight fluctuation only.

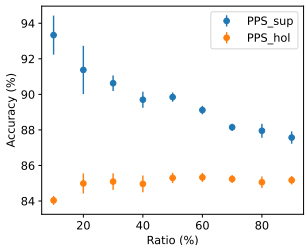
Layer importance. To confirm the contribution of layer importance, we run our LCPS on CIFAR-10-C in terms of Gaussian noise corruption. Fig. 6(c) displays the results of LCPS with and without layer importance. We notice that LCPS with layer weights requires fewer queries, i.e., converges faster, than LCPS without layer weights, showing a valuable property for the limited queries cases.

5 Related Work

Data privacy. Preventing adversarial inference of the data information during learning motivates our proposed setting. Recent study (Song et al., 2017) shows that the training data can be remembered by the deep model no matter it is accessed in a form of white-box or black-box. This alarms us that standard fine-tuning would expose data in a similar way if it was used by an attacker. Other work studies data privacy from different perspectives including cryptography (Chaum et al., 1988), differential privacy (Dwork et al., 2006), membership inference (Chen et al., 2020a), and federated learning (Shokri & Shmatikov, 2015). We focus on data extraction attack in this work, and leave other privacy concerns to future study.



(a) Batch size



(b) Support size

Error (%)	22.0	20.0
PPS	> 10.0	-
LCPS (w/o)	~ 3.0	> 8.4
LCPS	~ 0.8	~ 3.5

(c) Layer importance

Figure 6: Ablation study on three factors: batch size, support size, and layer importance (K queries).

Transfer learning. Our FTFB setting conceptually falls in transfer learning (Bengio, 2012) field from the learning paradigm as our goal is also transferring model for emerging test data. Domain adaptation (Quiñonero-Candela et al., 2009; Ganin & Lempitsky, 2015; Tzeng et al., 2015; Long et al., 2015), one of its trending subtopic, utilizes labelled source data to assist the learning of unlabelled target data, typically with a transductive learning manner. Recent works (Chidlovskii et al., 2016; Wang et al., 2020; Liang et al., 2020; Li et al., 2020a) further propose the setting of test-time adaptation which study how to adapt a pre-trained model to the test data free of source data, training loss, or even target labels. Our work is closely related to this thought, but they assume that test data features are at least observed, which does not satisfy the privacy need in our problem.

Black-box optimization: The technique we use to solve FTFB problem stems from the idea of black-box optimization. In black-box optimization, Bayesian optimization and derivative-free methods are widely used in hyper-parameter tuning (Turner et al., 2021) and black-box adversarial attacks (Ilyas et al., 2018). Our algorithms belong to the (gradient approximation) derivative-free regime. Directly applying derivative-free method (e.g., CMA-ES (Hansen, 2006)) in our application is not sample efficiency due to the high-dimensional nature of parameters in deep networks. Instead of treating the parameters of deep neural networks as a standard black-box optimization problem, we investigate the the layer-wise structure of the parameters. Our LCPS algorithm is applicable to the model parameters which are of higher dimension and even chained together.

6 Discussion

Summary. This paper presents a pioneer work of studying how to fine-tune a pre-trained model with only limited feedbacks to reduce the risk of data extraction proposed by Song et al. (2017). In this restrictive setting, data holders accept candidate models as queries and return the model performances as feedbacks. Compared with other related fine-tuning settings such as source-free UDA or FL, one can sense that the core thought of this setting is to preserve data privacy with the extra cost of communications between data holder and model provider. Therefore, to save the query budget, we do fine-tuning by characterizing the geometry of the objective w.r.t model parameters, which is inspired by black-box optimization technique. For more general cases where tuning parameters are high-dimensional and distributed across different layers of neural networks, we further propose a layerwise coordinate descent algorithm for practical usage. Extensive experimental results justify our setting and also show some potential applications, such as fair learning on private data, learning with indifferential metric, etc.

Future Works. (1) Our proposed optimization algorithms are developed on the idea of NES (Wierstra et al., 2014) which is actually not the unique choice for model parameters search. There are a number of surrogate objectives available from the literature (Vu et al., 2017). Recent study (Turner et al., 2021) finds out that the ensemble of some surrogates achieves superior results for the hyperparameter tuning task. Inspired by this finding, we will try to improve the query-efficiency of our algorithms in future work. (2) Apart from the data extraction risk, Membership Inference Attack (MIA) (Shokri et al., 2017), another prevalent type of the information leakage, also exists in the proposed setting of FTFB. Specifically, MIA tries to determine whether a specific instance was included in the target datas. Although this problem could be handled by Differential Privacy (DP) (Dwork et al., 2006) that aims to compensate the difference of the absence of every instance, a remaining challenge should be noticed, i.e., will the correlation of sequential queries in FTFB require a new composition theory for DP? We will explore this interesting question in the next work.

REFERENCES

Charles Audet and Warren Hare. Derivative-free and blackbox optimization. 2017.

Eugene Belilovsky, Michael Eickenberg, and Edouard Oyallon. Greedy layerwise learning can scale to imagenet. In *International conference on machine learning*, pp. 583–593. PMLR, 2019.

Yoshua Bengio. Deep learning of representations for unsupervised and transfer learning. In *Proceedings of ICML workshop on unsupervised and transfer learning*, pp. 17–36. JMLR Workshop and Conference Proceedings, 2012.

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012.

David Chaum, Claude Crépeau, and Ivan Damgard. Multiparty unconditionally secure protocols. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pp. 11–19, 1988.

Cen Chen, Bingzhe Wu, Minghui Qiu, Li Wang, and Jun Zhou. A comprehensive analysis of information leakage in deep transfer learning. *arXiv preprint arXiv:2009.01989*, 2020a.

Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. Big self-supervised models are strong semi-supervised learners. *arXiv preprint arXiv:2006.10029*, 2020b.

Boris Chidlovskii, Stéphane Clinchant, and Gabriela Csurka. Domain adaptation in the absence of source domain data. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 451–460, 2016.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pp. 4171–4186. Association for Computational Linguistics, 2019.

Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pp. 647–655. PMLR, 2014.

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pp. 265–284. Springer, 2006.

Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *International conference on machine learning*, pp. 1180–1189. PMLR, 2015.

John Geweke. Antithetic acceleration of monte carlo integration in bayesian inference. *Journal of Econometrics*, 38(1-2):73–89, 1988.

Nikolaus Hansen. The cma evolution strategy: a comparing review. *Towards a new evolutionary computation*, pp. 75–102, 2006.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019.

Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Black-box adversarial attacks with limited queries and information. In *International Conference on Machine Learning*, pp. 2137–2146. PMLR, 2018.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. PMLR, 2015.

Gwangseon Jang, Jin-woo Lee, Jae-Gil Lee, and Yunxin Liu. Distributed fine-tuning of cnns for image retrieval on multiple mobile devices. *Pervasive and Mobile Computing*, 64:101134, 2020.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114 (13):3521–3526, 2017.

Ron Kohavi. Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. In *Kdd*, volume 96, pp. 202–207, 1996.

Agustinus Kristiadi, Matthias Hein, and Philipp Hennig. Being bayesian, even just a bit, fixes overconfidence in relu networks. In *International Conference on Machine Learning*, pp. 5436–5446. PMLR, 2020.

Rui Li, Qianfen Jiao, Wenming Cao, Hau-San Wong, and Si Wu. Model adaptation: Unsupervised domain adaptation without source data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9641–9650, 2020a.

Yandong Li, Lijun Li, Liqiang Wang, Tong Zhang, and Boqing Gong. Nattack: Learning the distributions of adversarial examples for an improved black-box attack on deep neural networks. In *International Conference on Machine Learning*, pp. 3866–3876. PMLR, 2019.

Yanghao Li, Naiyan Wang, Jianping Shi, Jiaying Liu, and Xiaodi Hou. Revisiting batch normalization for practical domain adaptation. *arXiv preprint arXiv:1603.04779*, 2016.

Yijun Li, Richard Zhang, Jingwan Lu, and Eli Shechtman. Few-shot image generation with elastic weight consolidation. *arXiv preprint arXiv:2012.02780*, 2020b.

Jian Liang, Dapeng Hu, and Jiashi Feng. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In *International Conference on Machine Learning*, pp. 6028–6039. PMLR, 2020.

Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. Learning transferable features with deep adaptation networks. In *International conference on machine learning*, pp. 97–105. PMLR, 2015.

Julian McAuley and Jure Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*, pp. 165–172, 2013.

Yurii Nesterov and Vladimir Spokoiny. Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics*, 17(2):527–566, 2017.

Vadim Popov, Mikhail Kudinov, Irina Piontkovskaya, Petr Vytovtov, and Alex Nevidomsky. Distributed fine-tuning of language models on private data. In *International Conference on Learning Representations*, 2018.

Joaquin Quiñonero-Candela, Masashi Sugiyama, Neil D Lawrence, and Anton Schwaighofer. *Dataset shift in machine learning*. Mit Press, 2009.

Ilija Radosavovic, Justin Johnson, Saining Xie, Wan-Yen Lo, and Piotr Dollár. On network design spaces for visual recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1882–1890, 2019.

Roshni Sahoo, Divya Shanmugam, and John Guttag. Unsupervised domain adaptation in the absence of source data. *arXiv preprint arXiv:2007.10233*, 2020.

Yevgeny Seldin, Csaba Szepesvári, Peter Auer, and Yasin Abbasi-Yadkori. Evaluation and analysis of the performance of the exp3 algorithm in stochastic environments. In *European Workshop on Reinforcement Learning*, pp. 103–116. PMLR, 2013.

Gal Shachaf, Alon Brutzkus, and Amir Globerson. A theoretical analysis of fine-tuning with linear teachers. *arXiv preprint arXiv:2107.01641*, 2021.

Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pp. 1310–1321, 2015.

Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 3–18. IEEE, 2017.

Congzheng Song, Thomas Ristenpart, and Vitaly Shmatikov. Machine learning models that remember too much. In *Proceedings of the 2017 ACM SIGSAC Conference on computer and communications security*, pp. 587–601, 2017.

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei A Efros, and Moritz Hardt. Test-time training for out-of-distribution generalization. 2019.

Ryan Turner, David Eriksson, Michael McCourt, Juha Kiili, Eero Laaksonen, Zhen Xu, and Isabelle Guyon. Bayesian optimization is superior to random search for machine learning hyperparameter tuning: Analysis of the black-box optimization challenge 2020. *arXiv preprint arXiv:2104.10201*, 2021.

Eric Tzeng, Judy Hoffman, Trevor Darrell, and Kate Saenko. Simultaneous deep transfer across domains and tasks. In *Proceedings of the IEEE international conference on computer vision*, pp. 4068–4076, 2015.

Ky Khac Vu, Claudia d'Ambrosio, Youssef Hamadi, and Leo Liberti. Surrogate-based methods for black-box optimization. *International Transactions in Operational Research*, 24(3):393–424, 2017.

Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Fully test-time adaptation by entropy minimization. *arXiv preprint arXiv:2006.10726*, 2020.

Daan Wierstra, Tom Schaul, Tobias Glasmachers, Yi Sun, Jan Peters, and Jürgen Schmidhuber. Natural evolution strategies. *The Journal of Machine Learning Research*, 15(1):949–980, 2014.

Hongxu Yin, Arun Mallya, Arash Vahdat, Jose M Alvarez, Jan Kautz, and Pavlo Molchanov. See through gradients: Image batch recovery via gradinversion. *arXiv preprint arXiv:2104.07586*, 2021.

Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez Rodriguez, and Krishna P Gummadi. Fairness beyond disparate treatment & disparate impact: Learning classification without disparate mistreatment. In *Proceedings of the 26th international conference on world wide web*, pp. 1171–1180, 2017.

Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 14747–14756, 2019.

A More details about method

A.1 Gradient derivation of Eq. (2)

$$\begin{aligned}
\nabla_\omega J(\omega) &= \nabla_\omega \int E'(\mathcal{D};\theta)\pi(\theta|\omega)\, d\theta \\
&= \int E'(\mathcal{D};\theta)\nabla_\omega \pi(\theta|\omega)\, d\theta \\
&= \int E'(\mathcal{D};\theta)\frac{\pi(\theta|\omega)}{\pi(\theta|\omega)}\nabla_\omega \pi(\theta|\omega)\, d\theta \\
&= \int \pi(\theta|\omega)E'(\mathcal{D};\theta)\nabla_\omega \log \pi(\theta|\omega)\, d\theta \\
&= \mathbb{E}_{\pi(\theta|\omega)}[E'(\mathcal{D};\theta)\nabla_\omega \log(\pi(\theta|\omega))]
\end{aligned} \tag{7}$$

We assume that $\pi(\cdot)$ is a Gaussian distribution which takes advantages of exponential family. The gradient w.r.t. the variance therein typically demands more queries and more computation cost for the high-dimensional parameters, so we prefer the first-order optimization in this work.

A.2 TWO-PARTY SCENARIO EXPLANATION

In introduction, we claimed that the presented two-party scenario absorbs the special case where data holders execute tuning process themselves. This factually is supported by our algorithms introduced in Section 3. If data holders do not trust the provided model, they can exactly adopt the same way as used in two-party scenario, i.e., updating model by using their estimated gradients. This will avert the data extraction risk (Song et al., 2017) in both white and black model cases.

B MORE DETAILS ABOUT EXPERIMENTS

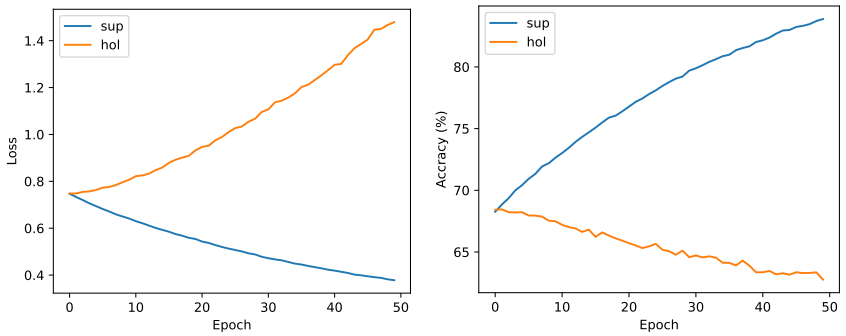
B.1 EXPERIMENTAL SETTING

Hyperparameters From the gradient based update of Eq. equation 4, the learning rate η and variance σ can be seen as an entirety. In experiments, we simply set $\frac{\eta}{\sigma} = 0.01$, which usually comes to the good results. Regarding crafting the candidate model, we typically set $\sigma = 0.1$ which leads to a moderate exploration throughout all experiments. The batch size b in our experiments is simply set as $\frac{d}{8}$ or $\frac{d}{16}$, in which the better result is reported in paper. We emphasize that this is only necessary if the query budget is tight, as they converge to the similar results. In Alg. 2, the learning rate β is simply set as 1.

We use the NVIDIA Quadro RTX 6000 Passive with 11,000M Memory (GPU) for all my experiments. Our project is based on public packages which are available for academic research under the license.

B.2 EVIDENCE OF FINE-TUNING ON PARTIAL PARAMETERS ONLY

We take the experiment for Amazon as an example, where the whole network is tuned with standard fine-tuning. We start with the good initial model, and the experimental results are shown as Fig. 8. The experiments demonstrate that with the optimization of standard fine-tuning, the tuning loss (on support set) significantly decreases while the generalization loss (on holdout set) increases dramatically. The corresponding accuracy also deviates from each other. Therefore, we conclude that when the whole network is fine-tuned for Amazon, the model tends to overfit on the support set only. We use this example to support our claim that in real applications the fine-tuning most time applies partial model parameters instead of the entire model.



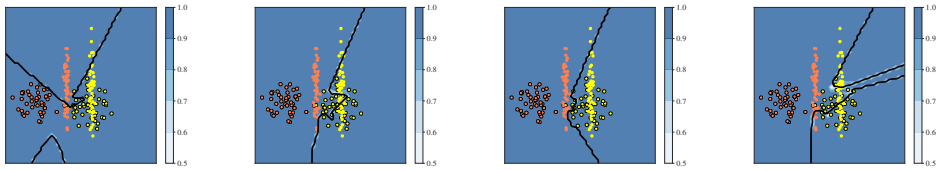
(a) Loss (b) Accuracy

Figure 7: Overfitting problem when tuning the whole network on Amazon.

B.3 ANTITHETIC SAMPLING

We verify the necessity of antithetic sampling in our toy examples. Without applying antithetic sampling, our algorithm PPS perform terribly on target data (see Fig. 8(a)). By increasing the

batch size, the performance improves but cannot rival the reported result in the main paper (Fig. 8(b&c)). We also try to use more queries but it turns out ineffective by Fig. 8(d). In particular, we notice the slight fluctuations always occur in decision boundary. In future work, we will explore this phenomenon from the view of robustness and variance control.



(a) (b) (c) (d)

Figure 8: Toy examples of FTFB without antithetic sampling for candidate model crafting. In (a)-(c), the batch size b is 80, 160, and 320, respectively, with the query budget of 8,000. In (d), the batch size is 80 with the query budget of 16,000.

B.4 Efficiency of updating all tuning parameters

For CIFAR-10-C, the network involves $8,064$ batch normalization parameters. Therefore, if we choose to fine-tune the model by updating all the model parameters, we basically query for 504 (i.e., $b = \frac{d}{16}$) times for only one step update. With one step update for all the parameters, we find that the updated model gets even worse than the pre-trained model sometimes, because it searches a large space and updates model once. By contrast, our layerwise algorithm updates model for 54 times with the same query budget, decreasing the error from 25.77% to 24.65% on support set.

VisDA-C. VisDA-C poses a challenging task of object recognition for 12 classes where the pre-trained model is built on synthesized data while the test data is collected from real scenes. Res-Net50 is employed and this group of experiments are performed based on the recent work (Liang et al., 2020) where the whole feature module is proposed to tune. Our experiments for object recognition on VisDa-C reaches the error of 45.5% on support set by only updating BN parameters, which is comparable with Tent (45.6%), while it needs around $4K$ queries.

B.5 Random Search

Random Search is previously used for hyperparamter tuning, which is demonstrated more efficient than grid search (Bergstra & Bengio, 2012). In our experiments, RS is used slightly different from the original paper. As model parameters are not explicitly bounded and we use "dynamic random search". Basically, every time we find some models which have better performances as seeds. Then we sample around them for multiple times and replace the seeds by more promising models. This thought is quite similar to evolutionary algorithm, but our experiments reveal it is not efficient for high-dimensional model parameters searching. Fig. 9 shows RS on CIFAR10-C with a random initialization. The tuning error (on support set) is around 85% at the beginning. By executing RS, the error is decreased and it eventually converges to 83%, showing an inefficient searching process in this challenging case.

B.6 Fair classification

In fair classification task, we fine-tune a pre-trained model to the target data where model accuracy and fairness are both required. We adopt discrimination parity as the fairness metric in this experiment. By setting the weight (ρ) of two measurements we update model as Alg. 3. $\rho = 0.8$ is used in our experiment.

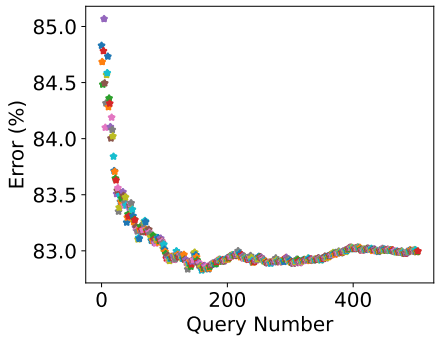


Figure 9: Random search results for high-dimensional parameters tuning.

Algorithm 3 Performance-guided Parameter Search (PPS) for Fair Classification

Input: variance σ, query budget q, batch size b, weight factor ρ
  for $t = 0, ..., \lfloor q/b \rfloor$ **do**
    Sample $\{\delta_j\}_{j=1}^{b/2} \sim \mathcal{N}(0, I)$, and for each j get $\delta_{b-j+1} = -\delta_j$
    Generate candidate models $\{\theta_i\}_i^b$ as queries where $\theta_i = \theta^t + \sigma\delta_i$
    Collect feedbacks $\{\text{Acc}(\mathcal{D}; \theta_i)\}_{i=1}^b$ and $\{\text{Dis}(\mathcal{D}; \theta_i)\}_{i=1}^b$
    $\theta^{t+1} \leftarrow \theta^t - \frac{\eta}{\sigma b} \sum_{i=1}^b \delta_i[\rho\text{Dis}(\mathcal{D}; \theta^t + \sigma\delta_i) - \text{Acc}(\mathcal{D}; \theta^t + \sigma\delta_i)]$
  end for
Output: $\theta^{\lfloor q/b \rfloor + 1}$