GSDYN: LEARNING TRAINING DYNAMICS VIA ONLINE GAUSSIAN OPTIMIZATION WITH GRADIENT STATES

Anonymous authors
Paper under double-blind review

ABSTRACT

Bayesian optimization, whose efficiency for automatic hyperparameter tuning has been verified over the decade, still faces a standing dilemma between massive consumption of time and suboptimal search results. Although much effort has been devoted to accelerate and improve the optimizer, the dominantly time-consuming step of evaluation receives relatively less attention. In this paper, we propose a novel online Bayesian algorithm, which optimizes hyperparameters and learns the training dynamics to make it free from the repeated complete evaluations. To solve the non-stationary problem i.e. the same hyperparameters will lead to varying results at different training steps, we combine the training loss and the dominant eigenvalue to track training dynamics. Compared to traditional algorithms, it saves time and utilizes the important intermediate information which are not well leveraged by classical Bayesian methods that only focus on the final results. The performance on CIFAR-10 and CIFAR-100 verifies the efficacy of our approach.

1 INTRODUCTION

Although deep learning has achieved great success across sundry tasks, especially empowered by the practical learning algorithms such as stochastic gradient descent (SGD), its generalization is still heavily dependent on the tuning of hyperparameters. Bayesian optimization (Snoek et al., 2012) has proven its superiority compared to baselines like random search or grid search but still faces enormous computational overhead. Many variants (Shahriari et al., 2016) have been proposed, which mostly focus on the optimizer while the dominantly time-consuming evaluation step still receives relatively less attention.

On the other hand, the vanilla Bayesian method (Snoek et al., 2012) only pays attention to human priors, and the final evaluations discard all the intermediate information. In contrast, in practice human experts prefer to track the training dynamics to tune the training process flexibly. The success of adaptive algorithms (Kingma & Ba, 2015) also suggests that effectively utilizing local information such as gradients, Hessian matrix etc., can lead to faster convergence and even helps achieve better generalization (Klein et al., 2016).

In this paper, we propose a novel online Bayesian optimization algorithm, which optimizes hyperparameters dynamically by fitting the non-stationary training process. From the standard Bayesian perspective, hidden distributions gradually get recognized by learning the black-box function between input hyperparameters and output observations with consecutive explorations and evaluations. However, as DNN's weights change during training, the same inputs will lead to various outputs. The black-box function will behave non-stationary, resulting in a phenomenon that the hidden distributions of inputs shift, which violates the Bayesian algorithm's basic assumption and hurts the prediction. In essence, the optimizer is misled by contradictory observations from diverse network states during training, while the actual distributions under a certain state should be consistent.

Thus, we aim to track the training dynamics and distinguish observations by adopting an online Bayesian inference procedure. Specifically, our proposed approach is called Gradient-State based learning Dynamics optimization (GSdyn). In the context of deep learning, one ideal but unrealistic way to track training dynamics is to memorize all the weights through training process. An alternative is to analyse the training trajectory on the loss surface. It has been discussed that SGD navigates the loss landscape through valleys (Xing et al., 2018) and tends to find a wide minima,

corresponding to a Hessian matrix with small dominant eigenvalues (Sagun et al., 2017), for better generalization (Hochreiter & Schmidhuber, 1997; Keskar et al., 2017; Wu et al., 2017). So we combine training loss and the dominant eigenvalue as state variables to help the surrogate model such as Gaussian Process(GP) (Rasmussen & Williams, 2005) to fit the training dynamics. As the momentum will suppress the spectrum norm's peak of dominant eigenvalues (Jastrzebski et al., 2019), GSdyn focuses on SGD without using momentum. We test our algorithm on CIFAR-10 and CIFAR-100 (Krizhevsky, 2009) with promising results.

In this paper, we address the problem for optimizing hyperparameters dynamically under the Bayesian framework over the training process (with an embodiment on deep neural network based image classification tasks). Empirical results show that our approach achieves better evaluation than manual tuning with a faster training convergence speed than the vanilla Bayesian method (Snoek et al., 2012) and its variants (Klein et al., 2016). Our contributions can be summarized as follows:

- To our best knowledge, our algorithm is one of the first works that can effectively optimize hyperparameters *on the fly*, in contrast to existing works that have to wait the completion of the overall evaluation in each step. Thus it significantly saves the time required for repeated and complete evaluations. Compared to standard methods, our algorithm also takes advantage of local information to search, leading to better generalization.

- We solve the unfitting problem through the non-stationary training process by tracking model states. It accelerates convergence and surpasses the performance under manual tuning on multiple public benchmark and empirically shows better generalization.

- Our work can also provide a new viewpoint for the Hessian spectrum, loss surface, etc, as Beyes is an excellent tool to construct a black-box function between variables. As shown in 5, GSdyn gains a good generalization while behaving to utilize the local information of the top dominant eigenvalues and control the learning rates away from too small.

2 Related Work

We review existing works in three aspects as follows, which we believe are most related to this paper.

Bayesian Optimization (BayesOpt): The classic Bayesian optimization framework (Snoek et al., 2012) involves a Gaussian process (Rasmussen & Williams, 2005) for quantifying the latent uncertainty and an acquisition function that samples new tops. Many efforts have been devoted to improving efficiency. The work (Swersky et al., 2013) proposes a multi-task Bayesian optimization method for hyperparameter learning and the training is accelerated by treating time cost as a variable in Bayesian inference. Based on the above work, the authors in (Klein et al., 2016) use the Bayesian method to search for the minimal training set's size needed through the optimization process to reduce the inner cost of the evaluation. The recent study (Falkner et al., 2018) tries to terminate the bad performers based on a multi-armed bandit developed in (Li et al., 2017), while in (Klein et al., 2017), it attempts to learn the learning curves as well as the timing for an early stop.

Tracking Hessian Spectrum: The Hessian matrix is essential for realizing the geometry of DNN's loss surface, whose spectrum norm has been tracked in the early work (LeCun et al., 1998). The work (Sagun et al., 2017) reports the distributions of all Hessian's eigenvalues and studies their relation with DNN's architecture and data. While in (Pennington & Bahri, 2017), a set of analytical tools are introduced from random matrix theory. In (Yao et al., 2018), the authors study the relationship between large batch size and loss surface based on the Hessian, and analyze its adversarial robustness. Moreover, it points out that the dominant eigenvalues are essential to guide the SGD step as characterizing the loss region's curvature (Jastrzebski et al., 2019).

SGD Dynamics The studies on SGD dynamics (Goodfellow et al., 2015; Chaudhari & Soatto, 2018) are also related to our work. The work (Xing et al., 2018) exposes the role and importance of a large learning rate through tracking SGD trajectory by interpolating the loss, whereas we find a related phenomenon that GSdyn will be stick in a *greedy trap* with a bad initialization of search space S. See Sec. 4.2 for more details.

Algorithm 1 Pseudo-code of Vanilla BayesOpt (Snoek et al., 2012)

1: **for** n=1,2,... **do**
2: Sample $\mathbf{x}_{n+1}$ by solving: $\mathbf{x}_{n+1} = \arg\max_{\mathbf{x}\in\mathcal{X}} a_p(\mathbf{x}; D_n)$;
3: Evaluate $y_{n+1} \sim f(\mathbf{x}_{n+1}) + N(0, \sigma^2)$ (the most expensive step);
4: Update $D_{n+1} = \{(\mathbf{x}_{n+1}, y_{n+1}), D_{n+1}\}$;
5: Refit D_{n+1} by $\mathcal{GP}$ (Rasmussen & Williams, 2005):

$$\mu_n(\mathbf{x}) = \mu_0(\mathbf{x}) + \mathbf{k}(\mathbf{x})^T(\mathbf{K} + \sigma_0^2\mathbf{I})^{-1}(\mathbf{y} - \mathbf{m})$$

$$\sigma_n^2 = k(\mathbf{x}, \mathbf{x}) - \mathbf{k}(\mathbf{x})^T(\mathbf{K} + \sigma^2\mathbf{I})^{-1}\mathbf{k}(\mathbf{x})$$

6: **end for**

3 PRELIMINARIES

In this section, we give some background information about our approach. Mathematically, given a black-box function $f\colon \mathbb{X} \to \mathbb{R}$, we are intersted in finding the $\mathbf{x}$ on a bounded set $\mathcal{X} \in \mathbb{R}^D$ that globally minimizes f:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}),$$

where D is the dimension of design space of interest and supposed to be lower than 20 (Frazier, 2018) in Bayesian optimization.

We construct the surrogate model $\mathcal{GP}$ based on a prior $p(f)$ and consecutive evaluations at a series of $\mathbf{x}$, sampled by an acquisition function $a_p(\mathbf{x})$. The fitting of the latent probability distribution will be updated as the observation increases. Summarily, as shown in Algorithm 1, the following three steps iterate (Klein et al., 2016; Snoek et al., 2012): 1) propose the most promising $\mathbf{x}_{n+1} \in \arg\max a_p(\mathbf{x})$, usually with MCMC methods (Gilks et al., 1997); 2) evaluate this recommendation to obtain a noisy result $y_{n+1} \sim f(\mathbf{x}_{n+1}) + N(0, \sigma^2)$ and augment the set $D_n = (\mathbf{x}_j, y_j)_{j=1,\dots,n}$ with $(\mathbf{x}_{n+1}, y_{n+1})$; 3) re-fit D to update $\mathcal{GP}$.

3.1 GAUSSIAN PROCESS

As a prominent choice of surrogate model for its analytical tractability and nice marginalization properties, Gaussian Process describes a multivariate Gaussian distribution on $\mathbb{R}^N$, determined by a mean function $m\colon \mathcal{X} \to \mathbb{R}$ and a positive definite covariance function $K\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. The covariance functions endow the Gaussian process with rich expression of black-box function f. The default choice is the ARD Matérn $5/2$ kernel (Snoek et al., 2012):

$$K_{\text{M52}}(\mathbf{x}, \mathbf{x}') = \theta_0 \left(1 + \sqrt{5d_\lambda(\mathbf{x}, \mathbf{x}')} + \frac{5}{3}d_\lambda(\mathbf{x}, \mathbf{x}')\right) \exp\left(-\sqrt{5d_\lambda(\mathbf{x}, \mathbf{x}')}\right),$$

where θ_0 and λ are free hyperparameters of $\mathcal{GP}$, and $d_\lambda(x, x') = (\mathbf{x} - \mathbf{x}')^\top \text{diag}(\lambda)(\mathbf{x}, \mathbf{x}')$ is the Mahalanobis distance. As the covariance function is crucial for determining how the $\mathcal{GP}$ fits observations, to show the efficiency of our algorithm itself, we simply initialize them with $\mathbf{1}$ in all our experiments.

3.2 ACQUISITION FUNCTIONS

The acquisition function $a_p\colon \mathcal{X} \to \mathbb{R}$ proposes the next $\mathbf{x}_{n+1}$ by examining all candidate points's utility for the next evaluation of f and returning the maximum, based on posterior induced from observations D_n via the surrogate model. There are always many choices of acquisition function including improvement-based, optimistic, and information-based policies, such as Probability Improvement (PI), Expectation Improvement (EI), Upper Confidence Bound (UCB), and Predictive Entropy Search (PES) (Shahriari et al., 2016).

In our experiments, we use EI for its robustness as has been well verified in many applications (Klein et al., 2016):

$$a_{\text{EI}}(\mathbf{x}, D_n) = \mathbb{E}_p\left[\max(f_{\min} - f(\mathbf{x}), 0)\right],$$

where $f_{\min}$ is the best function value known from D_n.

Algorithm 2 Pseudo-code of the proposed GSdyn

Input: Search space: S, Kernel parameters: $\kappa_0 = \{\mu_0, \sigma_0\}$, Network state: $\theta_{s0} = \{l_0, \lambda_0\}$, Gaussian process: $\mathcal{GP}$, Acquisition function: a_p, DNN model: M, Dataset for eigenvalues: D_e
Output: an over-parametrized network with a series of optimal $\mathbf{x}$;

Initialize $D_0 = \{(\mathbf{x}_{0i}, \theta_{s0}), y_{0i}\}_{i=1,2,...,T}$ using an initial design;
Compute θ_{s1} = Top_Eigenvalue (M, D_e);
for $n = 1, 2, ..., N$ **do**
    Refit κ_n = Update($\mathcal{GP}, D_{n-1}, \kappa_0$);
    Propose $\mathbf{x}_n$ = Sample($a_p, \kappa_n, S \times [\theta_{sn}, \theta_{sn}]$);
    Evaluate $y_n \sim f(\mathbf{x}_n) + N(0, \sigma^2)$ and update M;
    Compute θ_{sn} = Top_Eigenvalue (M, D_e);
    Augment $D_n = D_{n-1} \bigcup \{(\mathbf{x_n}, \theta_{sn}), y_n\}$;
end for

3.3 HESSIAN SPECTRUM

The Hessian matrix describes the curvature in loss surface, representing the local geometry combined with the loss itself. Based on these two state variables, GSdyn solves the non-stationary training dynamics fitting problem and make hyperparameters optimization online.

On the other hand, Hessian information is also helpful for training. It has been discussed that *wider minima* of over-parametrized neural networks, quantified by a low Hessian norm, should generalize better (Keskar et al., 2017; Dinh et al., 2017; Wen et al., 2018). The work (Xing et al., 2018) further argues that SGD bounces off walls and moves through valleys in loss surface, reaching a better generalization due to getting rid of sharp valleys.

Although it contains plentiful knowledge of the second-order information about the loss landscape, it is infeasible to compute the actual Hessian matrix. Based on the approximate method (Pearlmutter, 1994), the work (Sagun et al., 2017) reports that the spectral norm of the Hessian reduces through the training process, in which a bulk of eigenvalues centered around zero, and the rest few positives are discrete and away from zero. Meanwhile, there is a clear relationship that the bulk of eigenvalues depend on DNN's architecture. In contrast, the discrete set of large positive eigenvalues, called *dominant eigenvalues*, is related to training data.

Accordingly, the work (Jastrzebski et al., 2019) focuses on the dominant eigenvalues and studies the empirical correlation between the curvature and its generalization. It reports that the SGD tends to steer towards sharp regions of the loss surface directly in the beginning, leading to a rapid increase of dominant eigenvalues until the SGD step is too large to reduce the loss furthermore. It also suggests reducing the alignment of the SGD along the sharpest directions to accelerate the optimization.

Based on the above results, in this paper, we combine the training loss and the top dominant eigenvalue to capture the state of the network and learn local information. As the computation of the Hessian matrix and its eigenvalues is also time-consuming compared to DNN's training of several batches, we adopt the method of stochastic power iteration with acceleration (Sa et al., 2017) and only take a small subset of training data to compute. Our goal here is not to build a practical algorithm, but to explore online Bayesian optimization, which saves large evaluation overheads and takes advantage of local information through training dynamics. The details are described in Sec. 5.

4 THE PROPOSED APPROACH

Here, we introduce our new online Bayesian approach for Gradient State based learning Dynamics optimization (GSdyn). Compared to the vanilla Bayesian optimizers that model the loss or some other metrics of machine learning algorithms as a black-box function, and minimize it by consecutive evaluations, GSdyn evaluates hyperparameters over the training process on the fly, and adjusts them dynamically. Thus, GSdyn aims to learn the training dynamics and optimize it online, rather than finding an optimal $\mathbf{x} \in \mathcal{X}$ globally.

Along with the DNN's training, GSdyn gradually increases its familiarity with the network structure and training data, trying to balance the trade-off between exploration and exploitation under strict

constraints of the objective function. It implies that a new proposed hyperparameter selected could not always lead to a better result, but in the long run, it helps GSdyn learn training dynamics and how to optimize at different model states.

The pseudo-code of GSdyn is shown in Algorithm 2. It starts with an initial design and returns a raw dataset for the Bayesian inference. Then the acquisition functions iteratively propose new $\mathbf{x}$ after the surrogate model refits the increasing new observations. In our experiments, we take Bayesian inference at the end of every epoch. Besides the usual evaluations of the neural network, we also need to compute the training loss and the top dominant eigenvalue as state variables. Significantly, the state variables take part in the update of $\mathcal{GP}$, but should be isolated from the sampling of acquisition functions, i.e., the state variables are only used to assist $\mathcal{GP}$ in fitting non-stationary training dynamics. It plays no effect in DNN's training.

4.1 State kernel

To fix the non-stationary problem, the proposed GSdyn introduces $\theta = \{l, \lambda\}$ to represent model states, where l is the training loss and λ is the top dominant eigenvalue. Thus, the black-box function $f : \mathbb{X} \times \mathbb{R} \to \mathbb{R}$ need to take another two input dimensions. Note it should be limited to the current state when the acquisition functions draw new $\mathbf{x}$.

To extend the GP model by additional dimensions, we factorize the objective kernels by multiplying kernels (Swersky et al., 2013; Klein et al., 2016), consisting of the original kernel over hyperparameters and a extend state kernel:

$$K((\mathbf{x}, s), (\mathbf{x}', s')) = K_1(\mathbf{x}, \mathbf{x}') \cdot K_2(s, s')$$

Unfortunately, there is no much prior knowledge about distributions of the top dominant eigenvalue. Thus, we simply take the Matérn $5/2$ kernel (Snoek et al., 2012) and initialize their parameters to $\mathbf{1}$. The results in Sec. 5 demonstrate its effectiveness even by default kernels, which suggests the promising prospect of our approach.

4.2 Initialization design

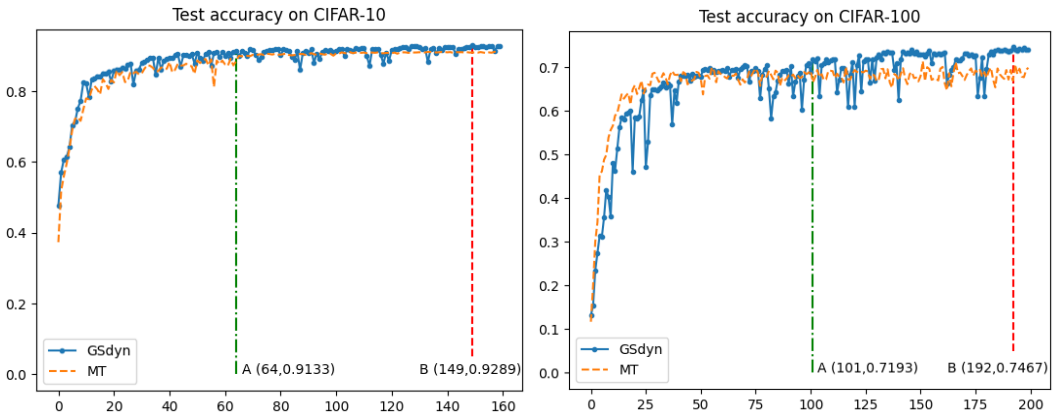
A general approach in Bayesian optimization is to take random search or some other methods to initialize the observation set at the beginning, as Bayesian algorithms perform poorly with uncertain prior and few evaluations. In our experiments, we adopt the Latin hypercube design to initialize the hyperparameters. To show the actual effectiveness of GSdyn, we evaluate these candidates by the same model only at the first epoch and choose the best one to update the DNN. Then through the whole training process, we implement the single candidate from the acquisition function to update the DNN directly, no matter how it performs. Similarly, we initialize the surrogate model simply with all $\mathbf{1}$.

The initialization of the search space S is crucial. Many works (Keskar et al., 2017; Smith, 2018; He et al., 2019) prove that small learning rates and large batch sizes always damage the final generalization. As Bayesian methods heavily depend on the observations and the prior, it prefers to propose $\mathbf{x}$ around the well-performed points. However, once the incumbent $\mathbf{x} \in S$ from random search only performs well in the beginning, it will take a long time to get rid of the *greedy trap*, especially using a small SGD step.

This phenomenon is also consistent with the experimental results in the work (Xing et al., 2018). It is reported that large learning rates maintains a large height of loss surface and assure SGD traverse larger distance, while SGD bounces off the walls of valleys.

4.3 Time Cost Analysis

The GSdyn's time consumption can be divided into three parts: the DNN's evaluations, the overheads of Bayesian inference, and the eigenvalue's computation. Compared with the vanilla Bayesian method (Snoek et al., 2012) and its variant (Klein et al., 2016), GSdyn substantially changes the mode of evaluations and save the plentiful cost of time. While the classical Bayesian methods need to wait for the completion of repeated evaluations, GSdyn can evaluate the DNN's performance at any time. The DNN with GSdyn only needs to be trained once.



(a) test accuracy on CIFAR-10 (b) test accuracy on CIFAR-100

Figure 1: Learning dynamics of GSdyn and Manual Tune (MT) on CIFAR-10 and CIFAR-100. The point A with a green line demonstrates when GSdyn surpasses MT's maximum at the end, while the point B with a red line demonstrates when GSdyn reach its own maximum.

Although the cost of Bayesian inference is notoriously heavy $O(N^3)$, it is relatively less considerable compared with the intractable cost of evaluations in vanilla Bayesian algorithms. GSdyn follow the same strategy that refit observations set every time and sample new candidate by EI with MCMC, to avoid prettifying its effectiveness. However, there are actually many improved methods (Kuzin et al., 2018; Zhang & Williamson, 2019) that accelerate inference.

Computing the actual Hessian is infeasible. Its time complexity is $O(D^2M)$ where D is the matrix rank and M is the number of matrix multiplications. Considering there are millions of parameters of DNN, we adopt the technique in (Sa et al., 2017) to simplify the time complexity to $O(D \cdot M)$ with a diagonal approximation (Becker & Lecun, 1989) of the Hessian, and reduce the amount of data used for computing the top dominant eigenvalues. There is still room for accelerating the calculation of the Hessian eigenvalues, which we leave for future work. On the other hand, compared with traditional Bayesian methods, the overheads of the top eigenvalue's computation is relatively cheaper.

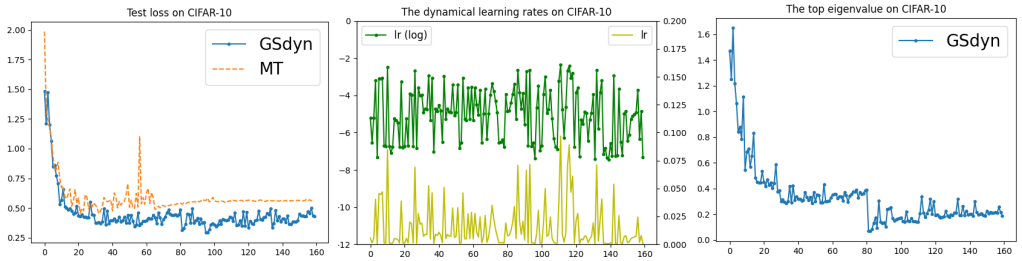
5 Experiments

The experiments are conducted on public benchmarks for deep network based image classification, to verify the cost-efficiency of our approach in comparison with peer methods. The source code will be made public available upon the final version of this paper.

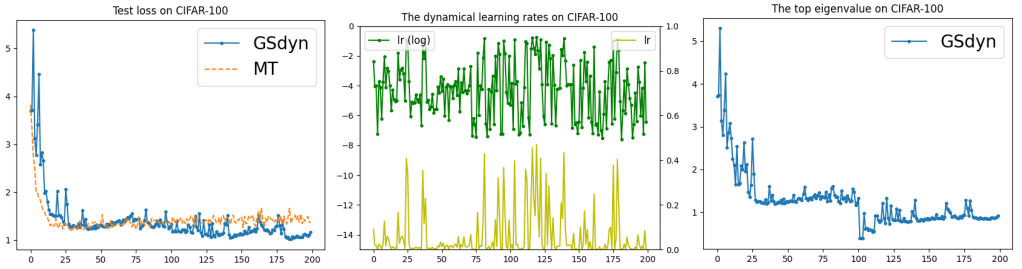
5.1 Protocols

Datasets. The CIFAR-10 dataset contains a total of 60,000 32*32 color images in 10 categories, each category has 6,000 images, lacking a pre-defined validation set. The CIFAR-100 dataset extends the categories to 100, having 600 images each (Krizhevsky, 2009). Thus, we split the trainset and use 40,000 images for training, 10,000 images to validate accuracy for Bayesian inference, and test the final performance on the standard 10,000 images. Meanwhile, the MT always trains on the whole 50,000 images and tests on the 10,000 images from testing set. For FABOLAS, we initialize its extra variable with a relative size of $\frac{1}{2}$, $\frac{1}{8}$ and $\frac{1}{16}$ for a comparison at similar time cost with total 50 steps of search.

Considering the time consumption of computing top dominant eigenvalue, we only select 5 ‰ of training data on both CIFAR-10 and CIFAR-100 so that the time required at each epoch was controlled to about 9 minutes. Note that it is reported that surpassing 5% of data samples can hardly lead to more benefits (Jastrzebski et al., 2019).



(a) test loss on CIFAR-10 (b) learning rate on CIFAR-10 (c) top eigen value on CIFAR-10



(d) test loss on CIFAR-100 (e) learning rate on CIFAR-100 (f) top eigen value on CIFAR-100

Figure 2: Test loss of GSdyn and MT (left), the dynamical learning rates tuned by GSdyn (middle) and the top dominant eigenvalue (right) through the whole training process on CIFAR-10 and 100.

Configurations and compared methods. We compare GSdyn with FABOLAS vanilla BO and manual tuning (MT) on CIFAR-10 and CIFAR-100 based on the same modified Resnet50 (He et al., 2016) using a single NVIDIA 2080Ti GPU. For the uniformity of the space, we suggest to take log transform on variables such as learning rates, accuracy.

We consider two hyperparameters: the learning rates (lr) and the weight decay (wd). We choose the same learning rates schedule in MT and FABOLAS as GSdyn tunes hyperparameters dynamically. Our partitions of dataset also keep the same across GSdyn and FABOLAS.

FABOLAS evaluates algorithms on subset of training data, determined by treating dataset size as an additional variable of Gaussian process. It is reported able to find good configurations 10 to 100 times faster than Hyperband (Li et al., 2017) and standard Bayesian optimization (Snoek et al., 2012). During the training process, we first draw 10 points of $\mathbf{x}$ proposed by random search and evaluate them on three subset with different size and then run Bayesian inference for 20 times, updating the surrogate model and sample new $\mathbf{x}$ overparameters and also s over dataset size repeatedly. The other parameters of the surrogate model including a customized linear Bayesian kernel are set as proposed in (Klein et al., 2016)

As the training loss has been used as the state variable, we choose DNN's accuracy as the objective function for both GSdyn and FABOLAS. Consider the relationship between the Hessian matrix and the neural network's generalization, we train the neural network only on the trainset and validate its accuracy on valset, while the baseline trains with all training data.

We initialize MT with $lr = 1e-1, wd = 1e-4$ and set batch size to 64 in all the experiments. The total number of training epochs is set to 160 for CIFAR-10 and 200 for CIFAR-100, respectively, across both GSdyn and FABOLAS. The schedule decreases the learning rate three times at $[65, 120, 160]$ with a $\gamma = 0.5$ for CIFAR-10 and at $[65, 100, 130]$ with $\gamma = 0.2$ for CIFAR-100.

5.2 Results and Discussion

The comparison is shown in Table 1. Note that the cost of vanilla BayesOpt is estimated based on the settings of FABOLAS.

Table 1: Comparison of Manual Tuning (MT), GSdyn and FABOLAS. The maximum* shows the cost of time and the epochs when the maximum is reached. For FABOLAS, the epoch means a complete evaluation of DNN. The cost of vanilla BO is estimated based on FABOLAS for comparison.

datasets	metrics	MT	GSdyn (ours)	FABOLAS	vanilla BO
CIFAR-10	total cost	3.4 h	28.8 h	27.6 h	170 h (estimated)
	accuracy (%)	91.12	**92.89**	88.98	
	maximum*	3.36 h / 158	26.8 h / **149**	27.6 h / 50	
CIFAR-100	total cost	4.3 h	36.7 h	35.3 h	215 h (estimated)
	accuracy (%)	71.43	**74.67**	63.92	
	maximum*	3.29 h / 192	36.23 h / **149**	21.18 h / 30	

As shown in Fig. 1, GSdyn can surpass MT using much fewer epochs and optimizes SGD to gain better generalization. Without accounting for all other overheads and focusing on the training process itself, we find it's novel that GSdyn accelerate the convergence almost 2x, while the only effect from GSdyn to DNN is adjusting hyperparameters. Meanwhile, the learning curve of GSdyn keeps undulating and gradually grows high, which shows that GSdyn balances the trade-off of exploration and exploitation. GSdyn can also revive the DNN more and more quickly with the increase of the observations, when the exploration causes a drop of accuracy. It implies that GSdyn can gradually learn the training dynamics and becomes an expert to realize how to tune hyperparameters at different model states.

More information is illustrated in Fig. 2. A clear correlation between generalization and the top dominant eigenvalue can be seen. We demonstrate the learning rates here and find that GSdyn does not minimize it as the training goes deeper. On the contrary, GSdyn selects hyperparameter points about evenly from the search space through the whole training process. Compared to the manual tuning which takes a multi-step reduction schedule, GSdyn gains a better generalization. This phenomenon is also consistent with the observation that the learning rates should not be too small as made in (He et al., 2019).

5.3 Further Discussion

Compared with the suboptimal performance of FABOLAS given the same approximately time cost as our GSdyn, the performance advantage of GSdyn is obvious. The traditional Bayesian algorithm is either time-consuming, or gives up part of information from the complete training to barter for accelerating obtain a speed bonus like Fast BO. Moreover, GSdyn not only tracks the indispensable training information, but also re-discovers more local information from training dynamics, helping it surpass FABOLAS in the specific training process. It also shows good generalization properties that surpasses manual tuning.

6 Conclusion and Outlook

In this paper, we have presented a novel online Bayesian algorithm for hyperparameter tuning, which optimizes hyperparameters and learns the training dynamics to avoid tedious evaluations from scratch. To solve the non-stationary issue i.e. the same hyperparameters will lead to varying results through the training process, we explore the training loss and the dominant eigenvalue to track training dynamics, in order to save time and utilizes the informative intermediate state which has not been well explored by classical Bayesian methods. The results on public benchmarks show the competitiveness and cost-efficiency of our approach.

The future work will involve more comprehensive evaluation across more datasets and more tasks beyond image classification, as well as improvements to the eigenvalue computing in terms of efficiency and accuracy.

REFERENCES

Sue Becker and Yann Lecun. Improving the convergence of back-propagation learning with second-order methods. pp. 29–37, 1989.

Pratik Chaudhari and Stefano Soatto. Stochastic gradient descent performs variational inference, converges to limit cycles for deep networks. In *2018 Information Theory and Applications Workshop (ITA)*, pp. 1–10, 2018.

Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. In *ICML'17 Proceedings of the 34th International Conference on Machine Learning - Volume 70*, pp. 1019–1028, 2017.

Stefan Falkner, Aaron Klein, and Frank Hutter. Bohb: Robust and efficient hyperparameter optimization at scale. *arXiv preprint arXiv:1807.01774*, 2018.

Peter I. Frazier. A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

W.R. Gilks, S. Richardson, and David Spiegelhalter. Markov chain monte carlo in practice. *Technometrics*, 39(3):338–338, 1997.

Ian J. Goodfellow, Oriol Vinyals, and Andrew M. Saxe. Qualitatively characterizing neural network optimization problems. In *ICLR 2015 : International Conference on Learning Representations 2015*, 2015.

Fengxiang He, Tongliang Liu, and Dacheng Tao. Control batch size and learning rate to generalize well: Theoretical and empirical evidence. In *NeurIPS 2019 : Thirty-third Conference on Neural Information Processing Systems*, pp. 1143–1152, 2019.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

Sepp Hochreiter and Jürgen Schmidhuber. Flat minima. *Neural Computation archive*, 9(1):1, 1997.

Stanislaw Jastrzebski, Zachary Kenton, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amos Storkey. On the relation between the sharpest directions of dnn loss and the sgd step length. In *ICLR 2019 : 7th International Conference on Learning Representations*, 2019.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *ICLR 2017 : International Conference on Learning Representations 2017*, 2017.

Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In *ICLR 2015 : International Conference on Learning Representations 2015*, 2015.

Aaron Klein, Stefan Falkner, Simon Bartels, Philipp Hennig, and Frank Hutter. Fast bayesian optimization of machine learning hyperparameters on large datasets. In *International Conference on Artificial Intelligence and Statistics (AISTATS 2017)*, pp. 528–536, 2016.

Aaron Klein, Stefan Falkner, Jost Tobias Springenberg, and Frank Hutter. Learning curve prediction with bayesian neural networks. In *ICLR 2017 : International Conference on Learning Representations 2017*, 2017.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

Danil Kuzin, Le Yang, Olga Isupova, and Lyudmila Mihaylova. Ensemble kalman filtering for online gaussian process regression and learning. In *2018 21st International Conference on Information Fusion (FUSION)*, pp. 39–46, 2018.

Yann LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural Networks: Tricks of the Trade, this book is an outgrowth of a 1996 NIPS workshop*, pp. 9–50, 1998.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: a novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18(1):6765–6816, 2017.

Barak A. Pearlmutter. Fast exact multiplication by the hessian. *Neural Computation*, 6(1):147–160, 1994.

Jeffrey Pennington and Yasaman Bahri. Geometry of neural network loss surfaces via random matrix theory. In *ICML'17 Proceedings of the 34th International Conference on Machine Learning - Volume 70*, pp. 2798–2806, 2017.

Carl Edward Rasmussen and Christopher K I Williams. *Gaussian Processes for Machine Learning*. 2005.

Christopher De Sa, Bryan D. He, Ioannis Mitliagkas, Christopher Ré, and Peng Xu. Accelerated stochastic power iteration. In *International Conference on Artificial Intelligence and Statistics*, volume 84, pp. 58–67, 2017.

Levent Sagun, Leon Bottou, and Yann LeCun. Eigenvalues of the hessian in deep learning: Singularity and beyond. *arXiv preprint arXiv:1611.07476*, 2017.

Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1): 148–175, 2016.

Leslie N. Smith. A disciplined approach to neural network hyper-parameters: Part 1 – learning rate, batch size, momentum, and weight decay. *arXiv preprint arXiv:1803.09820*, 2018.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems 25*, pp. 2951–2959, 2012.

Kevin Swersky, Jasper Snoek, and Ryan P Adams. Multi-task bayesian optimization. In *Advances in Neural Information Processing Systems 26*, pp. 2004–2012, 2013.

Wei Wen, Yandan Wang, Feng Yan, Cong Xu, Chunpeng Wu, Yiran Chen, and Hai Li. Smoothout: Smoothing out sharp minima to improve generalization in deep learning. *arXiv: Learning*, 2018.

Lei Wu, Zhanxing Zhu, and Weinan E. Towards understanding generalization of deep learning: Perspective of loss landscapes. *arXiv preprint arXiv:1706.10239*, 2017.

Chen Xing, Devansh Arpit, Christos Tsirigotis, and Yoshua Bengio. A walk with sgd. *arXiv preprint arXiv:1802.08770*, 2018.

Zhewei Yao, Amir Gholami, Kurt Keutzer, Michael W Mahoney, and Qi Lei. Hessian-based analysis of large batch training and robustness to adversaries. In *NIPS 2018: The 32nd Annual Conference on Neural Information Processing Systems*, pp. 4949–4959, 2018.

Michael Minyi Zhang and Sinead A. Williamson. Embarrassingly parallel inference for gaussian processes. *Journal of Machine Learning Research*, 20(169):1–26, 2019.