Learning Rate Perturbation: A Generic Plugin of Learning Rate schedule towards Flatter Local Minima

Hengyu Liu†
hengyuliu94@gmail.com
Northeastern University
Shenyang, Liaoning, China

Qiang Fu‡
qifu@microsoft.com
Microsoft Research Asia
Beijing, China

Lun Du
lun.du@microsoft.com
Microsoft Research Asia
Beijing, China

Tiancheng Zhang
{tczhang,yuge}@mail.neu.edu.cn
Northeastern University
Shenyang, Liaoning, China

Shi Han
{shihan,dongmeiz}@microsoft.com
Microsoft Research Asia
Beijing, China

ABSTRACT
Learning rate is one of the most important hyper-parameters that has significant influence for neural network training. Learning rate schedules are widely used in real practice to adjust the learning rate according to pre-defined schedules for the fast convergence and good generalization. However, existing learning rate schedules are all heuristic algorithms and lack theoretical support. Therefore, people usually choose the learning rate schedules through multiple ad-hoc trial, and the obtained learning rate schedules are sub-optimal. To boost the performance of the obtained sub-optimal learning rate schedule, we propose a generic learning rate schedule plugin, called LEArning Rate Perturbation (LEAP), which can be applied to various learning rate schedules to improve the model training by introducing a certain perturbation to the learning rate. We found that, with such simple yet effective strategy, training processing exponentially favors flat minima rather than sharp minima with guaranteed convergence, which leads to better generalization ability. In addition, we conduct extensive experiments which show that training with LEAP can improve the performance of various deep learning models on diverse datasets using various learning rate schedules (including constant learning rate).

CCS CONCEPTS
• Computing methodologies → Artificial intelligence; • Deep Learning → Neural Network.

KEYWORDS
deep learning, neural networks, learning rate scheduler

†Work done while this author was an intern at Microsoft Research
‡Corresponding Author

ACM Reference Format:
Hengyu Liu, Qiang Fu, Lun Du, Tiancheng Zhang, Ge Yu, Shi Han, and Dongmei Zhang. 2022. Learning Rate Perturbation: A Generic Plugin of Learning Rate schedule towards Flatter Local Minima. In Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM '22), October 17–21, 2022, Atlanta, GA, USA. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3511808.3557626

1 INTRODUCTION
Deep neural networks are the basis of state-of-the-art results for broad tasks such as image recognition, speech recognition, machine translation, driver-less car technology, source code understanding, social analysis, and tabular data understanding [6, 9, 21, 28, 29, 32]. It is known that the learning rate is one of the most important hyper-parameters which has significant influence for training deep neural networks. Too big learning rate may result in the great difficulty of finding minima or even the non-convergence issue, while too small learning rate greatly slows down the training process and may easily get stuck in sharp minima. How to adjust the learning rate is one of the challenges of training deep learning models.

Learning rate schedules seek to adjust the learning rate during training process by changing the learning rate according to a predefined schedule. Various learning rate schedules have been widely used and shown the practical effectiveness for better training. The existing learning rate schedules can be roughly divided into several categories, constant learning rate schedule, learning rate decay schedule[37], learning rate warm restart schedule [20, 22, 31] and adaptive learning rate schedule [17, 18, 24, 38]. However, the existing learning rate schedules are all heuristic algorithms and lack theoretical support. It is because that the mapping functions of DNN models are usually with high complexity and non-linearity, and it is quite difficult to analyze the influence of the learning rate on the training process. Therefore, people may choose a learning rate schedule only through multiple ad-hoc trial, and the obtained learning rate schedule are probably sub-optimal.

As an effective approach to improve generalization ability of DNN models, perturbation addition has been widely studied [25]. It has been demonstrated that training with noise can indeed lead to improvements in network generalization [2]. Existing works attempt to introduce perturbation in the feature space [1], activation function [10], and gradient [23]. [1] shows that adding noise to feature space during training is equivalent to Tikhonov regularization.
proposes to exploit the injection of appropriate perturbation to the layer activations so that the gradients may flow easily (i.e., mitigating the vanishing gradient problem). [23] adds perturbation to gradients to improve the robustness of the training process. The amount of noise can start high at the beginning of training and decrease over time, much like a decaying learning rate. However, there is no existing work that attempts to introduce randomness into hyper-parameters (e.g., learning rate).

To boost the obtained sub-optimal learning rate schedule, we propose a generic plugin of learning rate schedule, called Learning Rate Perturbation (LEAP), which can be applied to various learning rate schedules to improve the model training by introducing a certain type of perturbation to the learning rate. We leverage existing theoretical framework [34] to analyze the impact of LEAP on the training process, and found that training process applied with LEAP favors flatter minima exponentially more than sharpness-minima. It is well studied that learning flat minima closely relate to generalization.

In deep learning models, there are already some generic strategies or plugins that can boost training performance, such as Dropout, Batch Normalization. The reason why we consider them as a kind of plugins is they could be applied to a particular aspect of DNN learning with a wide range of versatility. In more details, Dropout could be considered as a generic plugin applied to the network structure of deep learning models, which prevents deep learning models from over-reliance on a single neural unit by randomly ignoring some weights during training processing. Batch Normalization is a generic plugin applied to layer input/output, which improves the performance of the model by normalizing the value range of intermediate layer input/output. With different perspective from Dropout and Batch Normalization, LEAP is a generic plugin applied to the learning rate, which improves model performance by letting the training process favor flat minima.

The main contributions of this work are outlined as follows:

- We propose a simple yet effective plugin of learning rate schedule, called LEA(rning) Rate Perturbation (LEAP), which can be applied to various learning rate schedules to improve the model training by introducing a certain perturbation to the learning rate.
- To the best of our knowledge, our work is the first one to propose a generic strategy with theoretical guarantees to boost training performance of various learning rate schedule by letting training process favor flat minima.
- The extensive experiments show that LEAP can effectively improve the training performance of various DNN architectures, including Multi-Layer Perceptron (MLP), Convolutional Neural Network (CNN), Graph Neural Network (GNN), and Transformer, with different learning rate schedules and optimizers on diverse domains.

2 LEARNING RATE PERTURBATION

In this section, we introduce the implementation details of LEAP, and given the pseudo code of training process applied with LEAP.

Firstly, we denote the training dataset as $x$, one batch of training datasets as $x_j$, the learning rate given by the learning rate schedule as $\eta$, the model parameters as $\theta$, and the loss function as $L(\theta, x)$. For simplicity, we denote the training loss as $L(\theta)$. Our LEAP is to add perturbation satisfying Gaussian distribution, i.e., $\zeta \sim N(0, \sigma^2)$, to the learning rate $\eta$. Here, $\sigma$ is the hyper-parameter of LEAP to control perturbation intensity. The training process applied with LEAP for updating parameters is shown in Algorithm 1.

Here, we define the learning rate after applying LEAP as a vector $h = (\eta_1, ..., \eta_M)$. $\eta_i$ is the learning rate used for updating $i$-th parameter, which is calculated by adding the learning rate perturbation $\zeta_i$ to the original learning rate $\eta$ given by the learning rate schedule, and $|h| = |\theta| = M$ (M is the total number of parameters). Note that, the perturbations on different feature dimensions ($\zeta_1, \zeta_2, ..., \zeta_M$) are independent so that the overall perturbations can lie along with sharpness-minima. It is well studied that learning flat minima closely relate to generalization.

The extensive experiments show that LEAP can effectively improve the training process applied with LEAP as follows.

$$
\theta_{t+1} = \theta_t - h \odot \frac{\partial L(\theta_t, x_j)}{\partial \theta_t} = \theta_t - \eta \frac{\partial L(\theta_t, x_j)}{\partial \theta_t} + \eta A(\theta) \zeta_t \tag{2}
$$

$$
A(\theta) = \text{diag}(\frac{\partial L(\theta_t, x_j)}{\partial \theta_{1,1}}, ..., \frac{\partial L(\theta_t, x_j)}{\partial \theta_{M, M}}) \tag{3}
$$

where $L(\theta, x_j)$ is the loss of the $j$-th mini-batch, $\odot$ is the Hadamard product, $\frac{\partial L(\theta_t, x_j)}{\partial \theta_{i,j}}$ is the gradient of the $i$-th parameter in the $j$-th update, and $\zeta_t \sim N(0, \sigma^2 I)$ which is a zero-mean high dimensional Gaussian distribution.

Algorithm 1: The training process applied with LEAP

| Input | training dataset $x$; learning rate schedule function $LRS$; init parameters $\theta$ |
|-------|--------------------------------------------------|
| Output| trained parameters $\theta$ |

1: for $e = 1$ to IterNum do
2: \quad $\eta_e = LRS(e)$
3: \quad for $j = 1$ to $B$ do
4: \quad \quad Get a batch of training data $x_j$ from training dataset
5: \quad \quad Sample $h \sim N(\eta_e \mathbf{A}, \eta^2 I)$
6: \quad \quad Update $\theta = \theta - h \odot \frac{\partial L(\theta, x_j)}{\partial \theta}$

3 MINIMA PREFERENCE ANALYSIS OF LEAP

By exploiting [34] theoretical framework, we replace stochastic gradient noise with $\eta A(\theta) \zeta_t$ and obtain Theorem 1. Theorem 1 present analysis result on escape time of LEAP at any minima $a$.

**Theorem 1.** LEAP Escape Time at Minima. The loss function $L(\theta)$ is of class $C^2$ and $N$-dimensional. If the dynamics is governed by equation 2, then the mean escape time from minima $a$ to the outside of minima $a$ is

$$
t = 2\pi C \frac{1}{|H_{ae}|} \exp \left[ \frac{2\Delta L}{\eta^2 s \frac{1}{A_{ae}}} \left( s + \frac{(1-s) s}{A_{ae}} \right) \right]
$$

where

$$
C = \sqrt{\frac{\text{det}(H_a - \text{diag}(H_a))}{\text{det}(H_a - \text{diag}(H_a))}}
$$

$s \in (0, 1)$ is a path-dependent parameter, $H_{ae}$ is the eigenvalues of Hessian matrix $H(\theta)$ at the saddle point $b$ corresponding to the escape direction $e$, and $A_{ae}$ and $\Delta L$.
Table 1: Error rate (%) of applying our LEAP to Cifar-10 dataset with two optimizers (lower indicators are better). The columns leaded by the cell "Gain" present relative improvements.

| Optimizer | Vanilla | LEAP (Ours) | Gain |
|-----------|---------|-------------|------|
| ResNet-18 | 5.60 ± 0.11 | 5.32 ± 0.25 | 5.00% |
| ResNet-50 | 5.26 ± 0.07 | 4.78 ± 0.13 | 9.13% |
| ResNet-101 | 5.09 ± 0.04 | 4.69 ± 0.08 | 7.86% |

| Optimizer | Vanilla | LEAP (Ours) | Gain |
|-----------|---------|-------------|------|
| Adam     | 9.26 ± 0.29 | 8.69 ± 0.08 | 6.16% |
| SGD      | 7.84 ± 0.52 | 7.17 ± 0.12 | 8.78% |
| Vanilla  | 7.13 ± 0.75 | 6.62 ± 0.04 | 7.15% |

Table 2: Error rate (%) of applying our LEAP to 5 MLP models, and 4 CNN models on three CV datasets without learning rate schedule (lower indicators are better). The columns leaded by the cell “Gain” present relative improvements.

| Dataset | Vanilla | LEAP (Ours) | Gain |
|---------|---------|-------------|------|
| MNIST MLP-3 | 2.24 ± 0.09 | 2.02 ± 0.07 | 9.82% |
| MNIST MLP-4 | 2.23 ± 0.06 | 2.04 ± 0.04 | 8.52% |
| MNIST MLP-6 | 2.24 ± 0.05 | 1.99 ± 0.13 | 11.16% |
| MNIST MLP-8 | 2.54 ± 0.09 | 2.26 ± 0.10 | 11.02% |
| MNIST MLP-10 | 2.44 ± 0.10 | 2.25 ± 0.9 | 7.79% |

| Dataset | Vanilla | LEAP (Ours) | Gain |
|---------|---------|-------------|------|
| Cifar-10 ResNet-18 | 7.11 ± 0.17 | 6.78 ± 0.06 | 4.64% |
| Cifar-10 ResNet-50 | 6.93 ± 0.10 | 6.57 ± 0.18 | 5.19% |
| Cifar-10 ResNet-101 | 6.67 ± 0.08 | 6.26 ± 0.20 | 6.15% |

| Dataset | Vanilla | LEAP (Ours) | Gain |
|---------|---------|-------------|------|
| IN ResNet-50 | 28.38 ± 0.09 | 26.74 ± 0.15 | 5.78% |
| IN VGG-19 | 31.17 ± 0.11 | 29.51 ± 0.17 | 5.33% |

Table 3: Error rate (%) of applying our LEAP to 4 GNN models on three Graph datasets without learning rate schedule (lower indicators are better). The columns leaded by the cell “Gain” present relative improvements.

| Dataset | Vanilla | LEAP (Ours) | Gain |
|---------|---------|-------------|------|
| Cora GCN | 25.38 ± 0.75 | 22.60 ± 0.57 | 10.95% |
| Cora GAT | 29.72 ± 1.21 | 26.18 ± 1.71 | 11.91% |
| Cora GIN | 37.46 ± 2.93 | 33.02 ± 2.71 | 11.85% |
| Cora GraphSage | 26.24 ± 0.45 | 23.50 ± 0.66 | 10.44% |

| Dataset | Vanilla | LEAP (Ours) | Gain |
|---------|---------|-------------|------|
| PubMed GCN | 25.96 ± 0.73 | 23.98 ± 1.00 | 7.63% |
| PubMed GAT | 26.60 ± 0.49 | 24.90 ± 1.63 | 6.39% |
| PubMed GIN | 29.24 ± 1.33 | 27.22 ± 1.87 | 6.91% |
| PubMed GraphSage | 27.02 ± 0.33 | 25.26 ± 1.26 | 6.51% |

| Dataset | Vanilla | LEAP (Ours) | Gain |
|---------|---------|-------------|------|
| CiteSeer GCN | 37.38 ± 1.36 | 33.68 ± 0.96 | 9.90% |
| CiteSeer GAT | 37.96 ± 3.65 | 35.36 ± 2.01 | 6.85% |
| CiteSeer GIN | 49.38 ± 3.30 | 46.02 ± 2.18 | 6.80% |
| CiteSeer GraphSage | 38.02 ± 0.90 | 34.64 ± 0.38 | 8.89% |

4 EXPERIMENT

4.1 Experiment Setup

Datasets. We take 6 real datasets from Computer Vision and Graph Learning domain to evaluate our method, and adopt Cora, Citeseer and PubMed [36] for Graph Learning domain [3, 5, 8], MNIST [16], CIFAR-10 [15] and ImageNet [4] (Denoted as IN in the experiment) for Computer Vision domain. Cora, Citeseer and PubMed [36] are citation networks based datasets. MNIST is one of the most researched datasets in machine learning, and is used to classify handwritten digits. The CIFAR-10 and ImageNet dataset is a collection of images that are commonly used to train machine learning and computer vision algorithms. For graph datasets, we use the public data splits provided by [36]. For the MNIST dataset, we keep the test set unchanged, and we randomly select 50,000 training images for training and the other 10,000 images as validation set for hyper-parameter tuning. For CIFAR-10 and ImageNet dataset, we use the public data splits provided by [15, 27].

Network Architecture. To verify the effectiveness of our method across different neural architectures, we employ Multilayer Perceptrons (MLPs), Convolutional Neural Networks (CNNs), Transforms, and Graph Neural Networks (GNNs) [7] for evaluation. And we adopt ResNet-18,50,101 [12], VGG-19 [30] for CNN, GCN [14], GAT [33], GIN [35], and GraphSage [11] for GNN, Swin Transformer [19] for Transforms. We run five MLP models which are denoted as MLP-L, L indicates the number of layers including input layer.

Learning Rate Schedules and Optimizers. We verify that LEAP on three different learning rate schedules (Constant Learning Rate, Learning Rate Decay and Warm Restart Schedule) [20]. In addition, we also explored the performance improvement of LEAP for two optimizers (SGD [26] and Adam [15]).

Hyperparameters Settings. For the models without classical hyperparameter settings, we performed a hyperparameter search to find the best one for the baseline model. And the hyperparameter search process ensure each model in the same domain has the same search space. For all the GNN models, learning rate is searched in [0.1, 0.05, 0.01, 5e-3, 1e-3], and weight decay is searched in [0.05, 0.01, 5e-3, 1e-3, 5e-4, 1e-4]. For ResNet and MLP, learning rate is searched in [0.1, 0.05, 0.01], and weight decay is searched in [5e-3, 1e-3, 5e-4, 1e-4]. Above is the hyperparameter setting.
We evaluate the effectiveness of our methods from four dimensions: (1) different learning rate schedules; (2) different neural architectures; (3) datasets from different domains; (4) different optimizers.

Under the setting without using learning rate schedule, i.e., equally to constant learning rate schedule, the error rates of the image classification task for the vanilla models with and without LEAP are shown in Table 2, and graph node classification task result is in Table 3. Under the setting with using learning rate schedule i.e., learning rate decay and warm restart schedule, the error rates of the image classification task are shown in Table 4, and graph node classification task result is in Table 5. Table 1 show that the error rates of the image classification task under different optimizers. Combining the results in Table 2, 3, 4, and 5, we can conclude that LEAP brings consistent improvements in these four dimensions. First, LEAP is a generic plugin, which can work with all learning rate schedules including Learning Rate Decay, and Warm Restart Schedule. Besides, we can see that LEAP can improve the performance of deep learning models even without basic learning rate schedules. We obtain 5.00 % to 13.65% relative error rate reduction with learning rate decay, and 6.65 % to 15.26% with Warm Restart Schedule. Second, LEAP brings consistent improvements across all neural architectures including MLPs, CNNs, Transformer, and GNNs. Third, vanilla models with LEAP have consistent improvements across both CV and Graph Learning domains. We obtain 4.64 % to 12.89% relative error rate reduction in CV domain, and 6.10 % to 15.26% in Graph domain. Last, optimizers does not affect the effectiveness of LEAP. SGD with momentum and Adam are top-2 widely-used optimizer, and Table 1 show that LEAP can achieve similar great results with both optimizers, which shows the significant practical potential of LEAP.

## 5 CONCLUSION

We propose a simple and effective learning rate schedule plugin, called LEA(r)ning Rate Perturbation (LEAP). Similar to Dropout and Batch Normalization, LEAP is a generic plugin for improving the performance of deep learning models. We found that LEAP make optimizers prefer flatter minima to sharp minima. Experiments show that training with LEAP improve the performance of various deep learning models on diverse datasets of different domains.

## 6 ACKNOWLEDGMENTS

Hengyu Liu’s work described in this paper was in part supported by the computing resources funded by National Natural Science Foundation of China (Nos.U1811261, 62137001).
