Improving Generalization Performance of Adaptive Learning Rate by Switching from Block Diagonal Matrix Preconditioning to SGD

Yasutoshi Ida
NTT Software Innovation Center
Tokyo, Japan
yasutoshi.ida@ieee.org

Yasuhiro Fujiwara
NTT Communication Science Laboratories
Tokyo, Japan
yasuhiro.fujiwara.kh@hco.ntt.co.jp

***Abstract*—Deep Neural Networks (DNNs) are widely used for various applications. Although adaptive learning rate algorithms are attractive for DNN training, their theoretical performance remains unclear. In fact, published analyses consider only simple optimization settings such as convex optimization, none of which are applicable to DNN training. This paper proposes TSO-ALRA, a two-stage optimizer using an adaptive learning rate algorithm; it is based on a full analysis of two approaches that do suit DNNs: parameter updates along geodesics on the statistical manifold and covariance structure of gradients. Our analysis reveals that the diagonal approximation used by existing adaptive learning rate algorithms inevitably degrades their efficiency. In addition, our analysis suggests that adaptive learning rate algorithms suffer drops in generalization performance in the last phase of training. To overcome these problems, TSO-ALRA combines an effective approximation technique and a switching strategy. Our experiments on several models and datasets show that TSO-ALRA efficiently converges with high generalization performance.**

I. Introduction

Deep Neural Networks (DNNs) are widely used for many applications such as image classification [1], [2], object detection [3], text classification [4] and artificial speech synthesis [5]. To train DNNs, Stochastic Gradient Descent (SGD) [6] remains a popular training algorithm. SGD uses the hyper-parameter called the *learning rate* to determine how much it updates parameters in DNNs during training. Since the learning rate largely determines the convergence speed and accuracy, it is one of the most important hyper-parameters for DNN training.

Adaptive learning rate algorithms are often used as they automatically adjust the learning rate for each dimension of the parameter. When we use plain training algorithms such as SGD, we typically set a scalar value as the learning rate, and this value is applied in common to each dimension of the parameter. On the other hand, since adaptive learning rate algorithms adjust the learning rate for each dimension of the parameter, they can effectively train models with parameters of high dimensionality such as DNNs [7]. The initial method in this line is AdaGrad [8]; it was proposed for convex optimization. In order to apply the idea to DNNs, several variants of AdaGrad such as RMSprop [9], Adadelta [10], Adam [11], SDProp [12] and AMSGrad [7] have been proposed, and they are widely used for DNN training.

However, the theoretical performance of adaptive learning rate algorithms in training complex models such as DNNs is unclear. The performance of AdaGrad, Adam and AMSGrad has been assessed only for convex optimization although DNN training involves non-convex optimization [7], [8], [11]. [13] found that the behavior of RMSProp is similar to an equilibration preconditioner. Since the equilibration preconditioner efficiently handles saddle points on the surface of the loss function with respect to parameters, RMSProp can efficiently train DNNs that have many saddle points [14]. However, RMSProp has a different formulation from the equilibration preconditioner. In addition, the reason for the similarity between RMSProp and the equilibration preconditioner is unclear.

In this paper, we propose TSO-ALRA, a two-stage optimizer using an adaptive learning rate algorithm for DNN training; it is founded on our analysis of the behaviors of existing adaptive learning rate algorithms. First, we investigate the updating rule of adaptive learning rate algorithms through an analysis of the geodesic on the statistical manifold. Intuitively, the geodesic is a straight line in a curved space. We reveal that an adaptive learning rate algorithm can efficiently train statistical models including DNNs if parameter updates are performed along a geodesic on the statistical manifold. However, standard adaptive learning rate algorithms such as Adam cannot realize this benefit due to their use of diagonal approximation. To overcome this problem, we introduce more accurate approximations than diagonal approximation such that the algorithm updates parameters along a geodesic. Next, we investigate the behavior of adaptive learning rate algorithms in the last phase of training. In particular, we analyze the behavior of diffusion around local minima via the covariance matrix on gradients. Although the plain SGD can escape from local minima via diffusion according to the covariance matrix of gradients [15], we reveal that adaptive learning rate algorithms swap the covariance matrix from an identity matrix around local minima. This result suggests that existing adaptive learning rate algorithms can be trapped in bad local minima yielding significant generalization errors. To deal with this problem, we use plain SGD in the last phase of training so as not to change the covariance structure. Our experiments show that TSO-ALRA efficiently converges with high generalization performance on famous models and several datasets.

II. Preliminary

In this section, we briefly review Stochastic Gradient Descent (SGD) and adaptive learning rate algorithms.

A. Stochastic Gradient Descent

Let us consider training dataset $Z = \{z_1, ..., z_N\}$. In the supervised setting, each sample is represented as $z_i = (x_i, y_i)$ where x_i and y_i are input and label data, respectively, and $z_i = x_i$ in the unsupervised setting. Many training algorithms minimize loss function $L_\theta(Z)$ with respect to parameter vector $\theta \in \mathcal{R}^P$ of the model [16], [17]. SGD is a popular training algorithm that uses samples from training dataset Z [6]. To minimize the loss function, SGD iteratively updates θ with a sample as follows:

$$\theta_{t+1} = \theta_t - \eta \nabla L_{\theta_t}(z_t), \tag{1}$$

where $\eta \in$ is the learning rate, $\theta_t \in \mathcal{R}^P$ is the parameter vector at time t, z_t is the sample at time t, and $\nabla L_{\theta_t}(z_t) \in \mathcal{R}^P$ is the first order gradient of the loss function with respect to the parameter given by z_t. When we use the mini-batch setting, we treat z_t as a subset of dataset Z.

B. Adaptive Learning Rate Algorithms

Adaptive learning rate algorithms efficiently train DNNs by adjusting the learning rate η in Equation (1) for each dimension of parameter vector θ_t. The full matrix version of AdaGrad [8] is a basic adaptive learning rate algorithm that updates parameters as follows:

$$\theta_{t+1} = \theta_t - \eta G_t^{-1/2} \nabla L_{\theta_t}(z_t), \tag{2}$$

where

$$G_t = \textstyle\sum_{i=1}^{t} [(\nabla L_{\theta_i}(z_i))(\nabla L_{\theta_i}(z_i))^T], \tag{3}$$

which is a $P \times P$ matrix. Since the size of P is large when we use large models such as DNNs, the following diagonal approximation is used for matrix G_t in Equation (2):

$$\theta_{t+1} = \theta_t - \eta \mathrm{diag}(G_t)^{-1/2} \nabla L_{\theta_t}(z_t). \tag{4}$$

RMSProp [9] changes this updating rule by using the following G_t instead of Equation (3):

$$G_t = \rho G_{t-1} + (1-\rho)(\nabla L_{\theta_t}(z_t))(\nabla L_{\theta_t}(z_t))^T, \tag{5}$$

where ρ is the decay rate used for computing the moving average. This heuristic offers higher convergence speed and accuracy compared than AdaGrad when we train DNNs [11]. AdaDelta [10], Adam [11] and AMSGrad [7] are variants of RMSProp.

These adaptive learning rate algorithms commonly use square roots of exponential moving averages of squared past gradients $G_t^{-1/2}$ to efficiently train models. The *regret analysis* can explain the role of $G_t^{-1/2}$ for convex optimization [8], [11], however, its role is unclear for non-convex loss functions such as those used in DNN training.

III. Proposed Method

In this section, we theoretically reveal two problems with existing adaptive learning rate algorithms for DNN training, and propose a two-stage optimizer using an adaptive learning rate algorithm, TSO-ALRA, that overcomes the problems. Our analysis focuses on the role of $G_t^{-1/2}$ because adaptive learning rate algorithms have $G_t^{-1/2}$ in the updating rule as in Equation (2) whereas plain SGD does not, see Equation (1).

First, we analyze the role of $G_t^{-1/2}$ for updating parameters by using the concept of geodesics on the statistical manifold. We show that an adaptive learning rate algorithm updates the parameter along a geodesic on the statistical manifold; the parameter can be effectively updated using this property. However, we also show that standard adaptive learning rate algorithms such as Adam suffer degraded effectiveness due to their use of diagonal approximation. To overcome this problem, we introduce more accurate approximations than diagonal approximations such as block diagonal approximation and Kronecker factored approximation.

Next, we investigate the behavior of diffusion around local minima via analysis of the covariance matrix on gradients. Although plain SGD can escape from local minima via the diffusion according to the covariance matrix of gradients [15], we show that adaptive learning rate algorithms break the covariance matrix; this property degrades the generalization performance because the algorithms are unable to escape from bad local minima triggering high generalization error. To deal with this problem, we use plain SGD in the last phase of training so that the covariance matrix does not break. Although the strategy is simple, we show that TSO-ALRA surprisingly improves the generalization performance in our experiments. Note that above concepts do not assume convex loss functions unlike previous works [8], [11].

A. Analysis of Geodesics

We analyze adaptive learning rate algorithms via geodesics on the statistical manifold. Before the analysis, we introduce several definitions. The following are definitions of statistical model and Fisher metric, which are popular concepts in the field of machine learning.

Definition 1 (Statistical Model). *Let us consider a family of probability distributions S. Suppose each element of S is a probability distribution $p_\theta \in S$ that is parametrized by $\theta \in \mathcal{R}^P$. We call such S a P-dimensional statistical model.*

Definition 2 (Fisher Metric). *Let S be a P-dimensional statistical model and $p_\theta(z)$ be a probability distribution of z. Given point θ, the Fisher metric of S at θ is defined as follows:*

$$F_\theta = \mathrm{E}_{p_\theta(z)}[(\nabla \log\, p_\theta(z))(\nabla \log\, p_\theta(z))^T], \tag{6}$$

where the size of F_θ is $P \times P$, $\mathrm{E}_{p_\theta(z)}[\cdot]$ denotes the expectation with respect to the distribution $p_\theta(z)$ and $\nabla = \frac{\partial}{\partial \theta}$.

The following is the definition of the statistical manifold, a central concept in Information Geometry [18].

Definition 3 (Statistical Manifold)**.** *For Riemannian manifold (S, F_θ), we say the manifold Statistical manifold S is equipped with Fisher metric F_θ. The tangent space at point p_θ on (S, F_θ) is denoted as $T_\theta S$. Metric F_θ induces the inner product in each tangent space $T_\theta S$: we denote the inner product of $u, v \in T_\theta S$ by $\langle u, v\rangle_{F_\theta} = u^T F_\theta v$. The norm of $u \in T_\theta S$ is defined as $||u||_{F_\theta} = \sqrt{\langle u, u\rangle_{F_\theta}}$.*

We define *geodesic* on a statistical manifold as follows:

Definition 4 (Geodesic)**.** *Let (S, F_θ) be a statistical manifold and $t \in \mathcal{R}$. The geodesic is defined as the constant speed curve $\theta(t) : t \to (S, F_\theta)$ that is locally distance-minimizing with respect to the distance induced by F_θ. The geodesic is represented as the following second order differential equation by using $\theta = \theta(t)$:*

$$\tfrac{d\theta_k}{dt^2} + \textstyle\sum_{i,j} \Gamma^k_{ij} \tfrac{d\theta_i}{dt}\tfrac{d\theta_j}{dt} = 0, \tag{7}$$

where

$$\Gamma^k_{ij} = \tfrac{1}{2}\textstyle\sum_l (F_\theta^{-1})_{kl}\big(\tfrac{\partial F_{\theta jl}}{\partial \theta_i} + \tfrac{\partial F_{\theta il}}{\partial \theta_j} - \tfrac{\partial F_{\theta ij}}{\partial \theta_l}\big). \tag{8}$$

Intuitively, the geodesic is a straight line in a curved space. In our case, the curved space is the statistical manifold. When we use negative log likelihood as a loss function such that $L_{\theta_t}(z_t) = -\log p_{\theta_t}(z_t)$, $G_t^{-1/2}$ in Equation (5) approximates $F_\theta^{-1/2}$ in Equation (6) as an online estimator. Therefore, we analyze the adaptive learning rate algorithm by investigating $F_\theta^{-1/2}$ instead of $G_t^{-1/2}$. Note that negative log likelihood is widely used as the loss function for DNN training. First, we have following property for $F_\theta^{-1/2}$:

Lemma 1. *Let us consider $u, v \in T_\theta S$ on (S, F_θ). Suppose that u and v are transformed into $F_\theta^{-1/2}u$ and $F_\theta^{-1/2}v$, respectively. Then, the inner product can written as $\langle F_\theta^{-1/2}u, F_\theta^{-1/2}v\rangle_{F_\theta} = u^T v$; and the norm becomes $||F_\theta^{-1/2}u||_{F_\theta} = \sqrt{u^T u}$.*

Proof. According to Definition 3, the inner product is given by $\langle F_\theta^{-1/2}u, F_\theta^{-1/2}v\rangle_{F_\theta} = (F_\theta^{-1/2}u)^T F_\theta F_\theta^{-1/2}v = u^T v$. In addition, the norm is given by $||F_\theta^{-1/2}u||_{F_\theta} = \sqrt{(F_\theta^{-1/2}u)^T F_\theta F_\theta^{-1/2}u} = \sqrt{u^T u}$. □

Notice that the inner product in Definition 3 is written as $u^T F_\theta v$ while in Lemma 1 it is $u^T v$. In other words, the tangent space of the statistical manifold is transformed into Euclidean space by using $F^{-1/2}$. In fact, Fisher metric F_θ is treated as identity matrix I, that is the metric of Euclidean space. The above yields the following lemma for the geodesic by using Lemma 1:

Lemma 2. *We assume that $u \in T_\theta S$ on (S, F_θ) and $t \in \mathcal{R}$. If we can transform u into $F_\theta^{-1/2}u$, the geodesic with direction of $F_\theta^{-1/2}u$ from θ can be written as $\theta + tF_\theta^{-1/2}u$.*

Proof. According to Lemma 1, the Fisher metric F_θ can be seen as identity matrix I by transforming u into $F_\theta^{-1/2}u$ as described in Lemma 2. In other words, the space can be seen as Euclidean space. In this case, since $\Gamma^k_{ij} = 0$ for Equation (8) by following [18], we have the following geodesic from Equation (7):

$$\tfrac{d\theta_k}{dt^2} = 0. \tag{9}$$

Then, we have the following solution for the above equation when the initial value and moving direction are $\theta(0) = \theta$ and $F_\theta^{-1/2}u$, respectively:

$$\theta_k(t) = \theta_k + t(F_\theta^{-1/2}u)_k, \tag{10}$$

where $(F_\theta^{-1/2}u)_k$ denotes the moving direction in the k dimension. Therefore, the geodesic with direction of $F_\theta^{-1/2}u$ from θ is given by $\theta + tF_\theta^{-1/2}u$. □

Lemma 2 reveals that geodesic $\theta(t)$ is the same as a straight line in Euclidean space if tangent vector u is transformed into $F_\theta^{-1/2}u$. Notice that in order to traverse a geodesic, we typically solve Equation (7), a second order differential equation. Thus, when we update parameters on a statistical manifold, the updating rules of training algorithms such as SGD must, in essence, consider Equation (7). On the other hand, Lemma 2 shows that we can use the more simple form of $\theta + tF_\theta^{-1/2}u$, rather than Equation (7), to traverse a geodesic. Thus we can derive a more efficient updating rule in terms of the geodesic by utilizing Lemma 2 as follows:

Theorem 1. *Let us consider statistical model p_θ and loss function $L(\theta)$ that is defined as a negative log likelihood. We assume that parameter θ is updated by an iterative algorithm such as SGD, and θ_t denotes the parameter at the t iteration. Then, we can update the parameter along the geodesic if we use the following updating rule:*

$$\theta_{t+1} = \theta_t - \eta F_{\theta_t}^{-1/2} g_t, \tag{11}$$

where $g_t = \nabla L(\theta_t)$ and η is the learning rate.

Proof. We obviously have Equation (11) by iteratively applying Lemma 2 to each θ_t. □

In addition, we have the following corollary by utilizing the relation of $G_t^{-1/2} \approx F_{\theta_t}^{-1/2}$ when we use negative log likelihood as the loss function:

Corollary 1. *If optimization is based on negative log likelihood, the adaptive learning rate algorithm updates the parameter along the geodesic by solving Equation (2).*

Corollary 1 shows that the adaptive learning rate algorithm yields efficient parameter updates because it can update the parameters along the geodesic while plain SGD cannot. Standard adaptive learning rate algorithms such as Adam and RMSProp approximate Equation (11) in Theorem 1 by using only the diagonal of F_{θ_t}. The updating rule with this diagonal approximation is given by $\theta_{t+1} = \theta_t - \eta\,\mathrm{diag}(F_{\theta_t})^{-1/2} g_t$ in RMPSrop. This formulation reduces the computation cost, however, it is difficult to apply Theorem 1 due to its diagonal approximation. In other words, it is difficult for standard adaptive learning rate algorithms to update parameters along a geodesic.

To overcome this problem, we use the following block diagonal approximation that is more accurate than the diagonal approximation:

$$\theta_{t+1} = \theta_t - \eta \tilde{F}_{\theta_t}^{-1/2} g_t, \tag{12}$$

and

$$\tilde{F}_{\theta_t} = \begin{bmatrix} F_{\theta_t}^{11} & 0 & \dots & 0 \\ 0 & F_{\theta_t}^{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & F_{\theta_t}^{LL} \end{bmatrix}, \tag{13}$$

where $\tilde{F}_{\theta_t}$ is a block diagonal matrix of Fisher metric, and $F_{\theta_t}^{ll}$ is the Fisher metric of the l DNN layer. Although $\tilde{F}_{\theta_t}$ ignores the correlation between layers, it handles all correlations within each layer. Since standard adaptive learning rate algorithms use Equation (4), which does not handle the correlation, our Equation (12) approximates Fisher metric more accurately. In terms of computation cost, if $W_l \in \mathcal{R}^{m\times n}$ is a parameter on the l layer and $F_{\theta_t}^{ll}$ is an $(mn) \times (mn)$ matrix, the computation cost of $(F_{\theta_t}^{ll})^{-1/2}$ is $\mathcal{O}(m^3n^3)$ because it needs Eigen-decomposition. In addition, as the elements of $F_{\theta_t}^{11}, ..., F_{\theta_t}^{LL}$ are independent of each other, we can effectively parallelize the computation of $\tilde{F}_{\theta_t}^{-1/2}$ as $(F_{\theta_t}^{11})^{-1/2}, ..., (F_{\theta_t}^{LL})^{-1/2}$.

Furthermore, we can reduce the computation cost from $\mathcal{O}(m^3n^3)$ to $\mathcal{O}(n^3 + m^3)$ for each $(F_{\theta_t}^{ll})^{-1/2}$ by utilizing Kronecker factored approximation [19]. Let $\alpha_l \in \mathcal{R}^m$ and $s_l \in \mathcal{R}^n$ be input activations and output of the l DNN layer, respectively. Then we have $s_l = W_l^T \alpha_l$. We assume that $\beta_l = \nabla L_{s_l}(z_t)$ which is computed by using back propagation. As a result, we have $\nabla L_{W_l}(z_t) = \alpha_t^l (\beta_t^l)^T$, i.e. the first order gradients of W_l. With this formula, we can approximate $F_{\theta_t}^{ll}$ by using Kronecker factored approximation as follows:

$$\begin{aligned} F_{\theta_t}^{ll} &= \mathrm{E}[g_t^l (g_t^l)^T] = \mathrm{E}[\beta_t^l (\beta_t^l)^T \otimes \alpha_t^l (\alpha_t^l)^T] \\ &\approx \mathrm{E}[\beta_t^l (\beta_t^l)^T] \otimes \mathrm{E}[\alpha_t^l (\alpha_t^l)^T] := B_t^l \otimes A_t^l := \bar{F}_{\theta_t}^{ll}, \end{aligned} \tag{14}$$

where $g_t^l = \mathrm{vec}(\nabla L_{W_l}(z_t))$, $\otimes$ is the operation of Kronecker product, and $\mathrm{vec}(\cdot)$ is the operation of vectorization. Then we can compute $(\bar{F}_{\theta_t}^{ll})^{-1/2} g_t^l$ where g_t^l represents the first order gradients for the l-th layer as follows:

$$\begin{aligned} (\bar{F}_{\theta_t}^{ll})^{-1/2} g_t^l &= \{B_t^l \otimes A_t^l\}^{-1/2} g_t^l \\ &= \{(B_t^l)^{-1/2} \otimes (A_t^l)^{-1/2}\} g_t^l \\ &= \mathrm{vec}((B_t^l)^{-1/2} g_t^l (A_t^l)^{-1/2}). \end{aligned} \tag{15}$$

The known identities of $(B_t^l \otimes A_t^l)^{-1/2} = (B_t^l)^{-1/2} \otimes (A_t^l)^{-1/2}$ and $((B_t^l)^{-1/2} \otimes (A_t^l)^{-1/2}) g_t^l = \mathrm{vec}((B_t^l)^{-1/2} g_t^l (A_t^l)^{-1/2})$ are used in the above equation. Finally, we have the following updating rule:

$$\begin{aligned} \theta_{t+1} &= \theta_t - \eta \begin{bmatrix} (\bar{F}_{\theta_t}^{11})^{-1/2} g_t^1 \\ \vdots \\ (\bar{F}_{\theta_t}^{LL})^{-1/2} g_t^L \end{bmatrix} \\ &= \theta_t - \eta \begin{bmatrix} \mathrm{vec}((B_t^1)^{-1/2} g_t^1 (A_t^1)^{-1/2}) \\ \vdots \\ \mathrm{vec}((B_t^L)^{-1/2} g_t^L (A_t^L)^{-1/2}) \end{bmatrix}. \end{aligned} \tag{16}$$

The computations of $(B_t^l)^{-1/2}$ and $(A_t^l)^{-1/2}$ in Equation (16) have costs of $\mathcal{O}(n^3)$ and $\mathcal{O}(m^3)$, respectively. Therefore, the total cost for the l-th layer is $\mathcal{O}(n^3 + m^3)$ as described above. The updating rule is more efficient than the original updating rule of Equation (11) because it has cost of $\mathcal{O}(P^3)$. Since usually $P >> m, n$, we can effectively reduce the cost by using the above block diagonal approximation and Kronecker factored approximation. In addition, we can effectively parallelize the computation because B_t^l and A_t^l can be computed simultaneously. Note that since B_t^l and A_t^l are positive-semidefinite matrices in the definition of Equation (14), all eigen values are greater than or equal to zero. Thus we can safely compute the square root of the matrices $(B_t^l)^{-1/2}$ and $(A_t^l)^{-1/2}$ by combining damping technique [19] and Eigen-decomposition such as $(B_t^l)^{-1/2} = U(\Lambda + \epsilon I)^{-1/2} U^T$ where ϵ is a small constant value such as 10^{-6}, U and Λ are eigen vectors and eigen values of B_t^l, respectively.

To compute B_t^l and A_t^l in Equation (14), we use exponential moving averages as follows:

$$\bar{B}_t^l = (1-\gamma)\bar{B}_{t-1}^l + \gamma \beta_t^l (\beta_t^l)^T \tag{17}$$

$$\bar{A}_t^l = (1-\gamma)\bar{A}_{t-1}^l + \gamma \alpha_t^l (\alpha_t^l)^T \tag{18}$$

where $\bar{B}_t^l$ and $\bar{A}_t^l$ are estimated matrices of B_t^l and A_t^l, respectively. $\gamma \in [0, 1)$ is a hyper-parameter for computing the exponential moving averages. In addition, we update $(B_t^l)^{-1/2}$ and $(A_t^l)^{-1/2}$ in Equation (16) for every T_{int} step to reduce the number of computations of Eigen-decomposition following [19].

B. Analysis of Covariance Structure

In this section, we show that the formulation of $F_{\theta_t}^{-1/2} g_t$ can negatively impact generalization performance. In particular, the behavior of diffusion around local minima is isotropic due to $F_{\theta_t}^{-1/2} g_t$. Since the plain SGD can escape from local minima via anisotropic diffusion [15], our result suggests that existing adaptive learning rate algorithms can be trapped in bad local minima yielding significant generalization error. In our analysis, we assume the following Gaussian distribution on the first order gradient around local minima, see [12], [20]:

$$g_t \sim N\big(\bar{g}_t, \tfrac{1}{b} C_t\big), \tag{19}$$

where $g_t := \nabla L_{\theta_t}(z_t)$, $\bar{g}_t$ is the full gradient at the t iteration that is computed by using all data points, b is the size of the mini-batch, and

$$C_t = \rho C_{t-1} + (1-\rho)(g_t - \bar{g}_t)(g_t - \bar{g}_t)^T. \tag{20}$$

This is, in effect, an online estimator for the covariance matrix of first order gradients. [20] discuss the validity of this assumption near a local minimum. Since we are investigating the covariance structure of first order gradients around local minima, we use an assumption that will hold in the final phase of SGD. According to the assumption, we derive the following lemma:

Lemma 3. *Let us consider local minimum θ^*. If $C(\theta^*)$ is a covariance matrix around θ^* that is computed by using Equation (20), we have the following equation:*

$$\mathrm{E}[g(\theta^*)g(\theta^*)^T] \approx C(\theta^*) \approx G(\theta^*), \tag{21}$$

where $g(\theta^*)$ is the gradient around θ^*, and $G(\theta^*)$ is computed around θ^* by using Equation (5).

Proof. The covariance matrix around local minima is $\mathrm{E}[g(\theta^*)g(\theta^*)^T]$ by following [15]. Since the covariance matrix is approximated by using the online estimator of Equation (20), we have $\mathrm{E}[g(\theta^*)g(\theta^*)^T] \approx C(\theta^*)$. In addition, since $g(\theta^*) = \nabla L_{\theta^*}(z)$, we have $\mathrm{E}[g(\theta^*)g(\theta^*)^T] = \mathrm{E}[\nabla L_{\theta^*}(z)\nabla L_{\theta^*}(z)^T] \approx G(\theta^*)$ by using the online estimator of Equation (5). As the result, we have $\mathrm{E}[g(\theta^*)g(\theta^*)^T] \approx C(\theta^*) \approx G(\theta^*)$ as shown in Equation (21). □

Lemma 3 shows that the result of Equation (5) is the same as Equation (20) around local minima. Based on Lemma 3, we have the following theorem for the adaptive learning rate algorithm around local minima:

Theorem 2. *Let us consider an adaptive learning rate algorithm that updates parameters by using Equation (2) and (5). Then we have the following Gaussian distribution on $G_t^{-1/2}g_t$ around local minimum θ^*:*

$$G_t^{-1/2}g_t \sim N\big(G(\theta^*)^{-1/2}\bar{g}_t, \tfrac{1}{b}I\big). \tag{22}$$

Proof. According to Equation (19) and Equation (21) in Lemma 3, the Gaussian distribution can be approximated around local minimum θ^* as follows:

$$g_t \sim N\big(\bar{g}_t, \tfrac{1}{b}G(\theta^*)\big). \tag{23}$$

Since $G_t = G(\theta^*)$ around local minimum θ^*, we have Equation (22) as follows:

$$\begin{aligned} G_t^{-\frac{1}{2}}g_t &\sim N\big(G(\theta^*)^{-\frac{1}{2}}\bar{g}_t, \tfrac{1}{b}G(\theta^*)^{-\frac{1}{2}}G(\theta^*)(G(\theta^*)^{-\frac{1}{2}})^T\big) \\ &= N\big(G(\theta^*)^{-1/2}\bar{g}_t, \tfrac{1}{b}I\big). \end{aligned}$$

□

In addition, since $G_t^{-1/2} \approx F_{\theta_t}^{-1/2}$ when we use negative log likelihood as the loss function, we have the following corollary:

Corollary 2. *When we optimize a negative log likelihood, we have the following Gaussian distribution instead of Equation (22) in Theorem 2:*

$$G_t^{-1/2}g_t \approx F_{\theta_t}^{-1/2}g_t \sim N\big(F_{\theta^*}^{-1/2}\bar{g}_t, \tfrac{1}{b}I\big). \tag{24}$$

Algorithm 1 TSO-ALRA.

```
Require: update interval $T_{int}$, hyper-parameter of exponential
    moving average $\gamma \in [0,1)$, learning rate $\eta$
Require: $t \leftarrow 0$, $s \leftarrow 0$, initialize $\{B_t^l\}_{l=1}^L$ and $\{A_t^l\}_{l=1}^L$
 1: while stopping criterion not met do
 2:   $t \leftarrow t+1$
 3:   Update $\{B_t^l\}_{l=1}^L$ and $\{A_t^l\}_{l=1}^L$ by using Equation (17)
      and (18)
 4:   if $t \equiv 0$ (mod $T_{int}$) then
 5:     Update $\{(B_t^l)^{-1/2}\}_{l=1}^L$ and $\{(A_t^l)^{-1/2}\}_{l=1}^L$ by using
        Eigen-decomposition
 6:     $s \leftarrow s+1$
 7:   end if
 8:   Update $\theta_t$ by using Equation (16), $\{(B_{s\cdot T_{int}}^l)^{-1/2}\}_{l=1}^L$
      and $\{(A_{s\cdot T_{int}}^l)^{-1/2}\}_{l=1}^L$
 9: end while
10: while stopping criterion not met do
11:   $t \leftarrow t+1$
12:   Update $\theta_t$ by using Equation (1)
13: end while
```

The theorem and corollary show that covariance matrix C_t is diagonalized around local minima by using $F_{\theta_t}^{-1/2}g_t$ (or $G_t^{-1/2}g_t$) instead of g_t. The plain SGD is diffuse according to covariance matrix C_t, and can escape from bad local minima that have high generalization error as shown in [15]. However, the diffusion of adaptive learning rate algorithm is isotropic if it uses $F_{\theta_t}^{-1/2}g_t$ (or $G_t^{-1/2}g_t$) instead of g_t because the covariance matrix C_t changes to identity matrix I in Theorem 2 and Corollary 2. The result suggests that it is difficult to escape from bad local minima (which yield high generalization error) when we use conventional adaptive learning rate algorithms.

To deal with the above problem, we apply plain SGD rather than the conventional adaptive learning rate algorithm in the last phase of training so that the covariance matrix does not turn into an identity matrix. Although we determine the switchover point as a hyper-parameter by using validation data, the same as the standard approach in machine learning, we can also determine it by monitoring validation loss in real-time, which is often necessary in deep learning. This real-time strategy can be used for determining hyper-parameters such as the decay schedule of the learning rate. In a similar way, we can monitor the validation loss in each epoch, and can replace the adaptive learning rate algorithm with plain SGD when the minimum validation loss is not altered over several epochs.

C. Algorithm

We summarize TSO-ALRA as the pseudo code in Algorithm 1. First, we compute B_t^l and A_t^l for each layer by using Equation (17) and (18) (line 3). In Equation (18), α_t^l is computed from the result of forward propagation. Similarly, β_t^l in Equation (17) is computed from the result of back propagation. Next, we compute $(B_t^l)^{-1/2}$ and $(A_t^l)^{-1/2}$ for every T_{int} step (line 4-7). The computations can be made by using Eigen-decomposition as in $(B^l_t)^{-1/2} = U\Lambda^{-1/2}U^T$. Note that the computations of $(B^l_t)^{-1/2}$ and $(A^l_t)^{-1/2}$ can be parallelized. In addition, they can be parallelized for each layer. Then, we update parameter θ_t by using Equation (16) (line 8). Equation (16) effectively updates the parameter along a geodesic as we proved in Theorem 1 and Corollary 1. Since we update $(B^l_t)^{-1/2}$ and $(A^l_t)^{-1/2}$ for every T_{int} step, we use $(B^l_{s\cdot T_{int}})^{-1/2}$ and $(A^l_{s\cdot T_{int}})^{-1/2}$ in the equation. If a stopping criterion is satisfied (line 1), we use plain SGD for the training (line 11) until convergence. The stopping criterion is defined by the user. Although we determine the switchover point by using validation loss, the same as the standard approach for determining hyper-parameters in machine learning, we can also employ other rules for determining it as described in the previous section. As shown by Corollary 2, we can theoretically avoid breaking the covariance structure of first order gradients by using plain SGD in the last phase of training, and improve the generalization performance as the diffusion of plain SGD helps the process to escape from local minima.

IV. Related Work

We introduce previous works in three areas related to our work: adaptive learning rate algorithms, natural gradient and switching strategy.

Adaptive learning rate algorithms. Before the Deep Learning Era, adaptive learning rate algorithms were used to accelerate the training of neural networks. There are two approaches; adapting a global learning rate and local learning rates. Global learning rate is shared on all parameters. The papers of [21]–[23] adapt global learning rate with the full-batch setting. The on-line version of the approach was proposed in [24]. On the other hand, [25] adapt local learning rate that applies different learning rates to each parameter. [26] proposed an on-line version of this approach.

In the Deep Learning Era, several adaptive learning rate algorithms such as RMSProp [9], AdaDelta [10] and Adam [11] were proposed; all were based on AdaGrad [8] an optimization method for convex settings. Adam has been successful in a wide range of applications (over 10,000 citations), and several follow-up methods such as AMSGrad [7] have been proposed. These methods have low computation costs so that they can train DNNs even though they have large numbers of parameters. However, the performance of these methods was proved using convex settings which is not possible with DNNs, and only empirical demonstrations on DNNs have been published. In addition, some researchers have reported that Adam does not converge to the optimal solution [7], and in fact its generalization performance is worse than that of plain SGD [27]. On the other hand, our analysis that focuses on geodesics holds for DNNs with negative log likelihood cost functions. Furthermore, since TSO-ALRA replaces the adaptive learning rate algorithm with plain SGD in the last phase of training, it offers superior generalization performance to Adam.

Natural gradient. Natural Gradient has been used as an effective training algorithm for neural networks [28], and recently for DNN training [19], [29], [30]. It uses Fisher Information Matrix F_θ as the metric while the adaptive learning algorithm uses $F_\theta^{1/2}$. Natural Gradient computes the steepest descent on the statistical manifold by using F_θ. However, the role of $F_\theta^{1/2}$ in adaptive learning rate algorithms was unclear for a long time as pointed out in [31]. Our analysis reveals this role through Corollary 1; adaptive learning rate algorithms update parameters along a geodesic by using $F_\theta^{1/2}$.

Switching strategy. The strategy of replacing adaptive learning rate algorithm with plain SGD has been used in the practical training of neural networks such as Google's Neural Machine Translation System [32]. They showed that this switching strategy improves the speed of convergence and generalization performance. [33] proposed a switching strategy that automatically determines the switchover point. [34] use a similar strategy for training with large mini-batch; it smoothly switches from the adaptive learning rate algorithm to plain SGD. The above works have empirically confirm the effectiveness of the switching strategy. Our analysis provides a theoretical understanding of the effectiveness through Theorem 1, 2, Corollary 1 and 2.

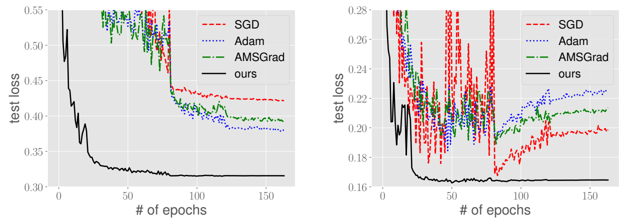
V. Experiments

We performed image classification experiments to confirm that TSO-ALRA offers better (i) convergence speed and (ii) generalization performance (test error) than other adaptive learning rate algorithms.

A. Setting

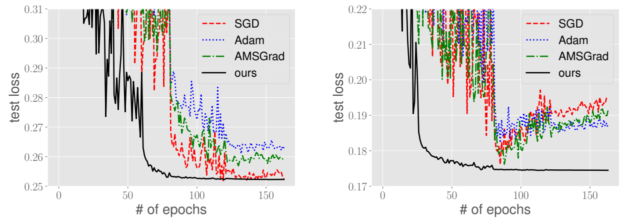
We used four various datasets to investigate TSO-ALRA: SVHN [35], Fashion-MNIST [36], Cifar-10 and Cifar-100 [37]. SVHN is the Street View House Numbers dataset that has 10-class color house-number images from the real-world. Fashion-MNIST contains 10-class gray scale images of fashion products; image classification is harder to achieve than standard the MNIST dataset. Cifar-10 and Cifar-100 are color images datasets of various objects that have 10 and 100 classes, respectively. We divided each of the above datasets into training, validation and test data; validation data is used for determining hyper-parameters such as learning rate. We used two famous models; AlexNet [1] and VGG [38]. AlexNet and VGG are famous solutions proven in the competitions of ILSVRC 2012 and 2014, respectively. The hyper-parameters of the models such as kernel size and stride were modified to fit the image sizes of each dataset.

The baseline algorithms are plain SGD, Adam, and AMSGrad. Adam is a variant of RMSProp; it integrates momentum into RMSProp. AMSGrad is a state-of-the-art algorithm based on Adam. For Adam and AMSGrad, we tuned the hyper-parameters on $\{0.9, 0.99\}$ using the coefficient of momentum, and the learning rates of $\{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$. We also tuned the learning rate of plain SGD using $\{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$. In TSO-ALRA, the learning rate and switchover point were tuned using $\{10^{-1}, 10^{-2}, 10^{-3}\}$, and $\{5, 10, ..., 70, 75\}$ epochs, respectively. In TSO-ALRA, the learning rate was tuned using $\{10^{-1}, 10^{-2}, 10^{-3}\}$. As a result,



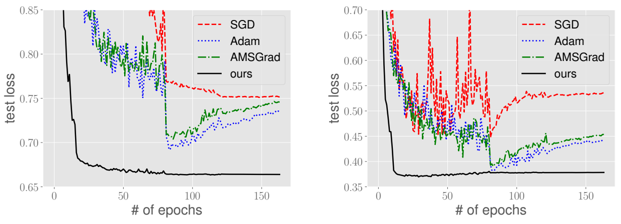
(a) AlexNet (b) VGG

Fig. 1. Test loss on SVHN



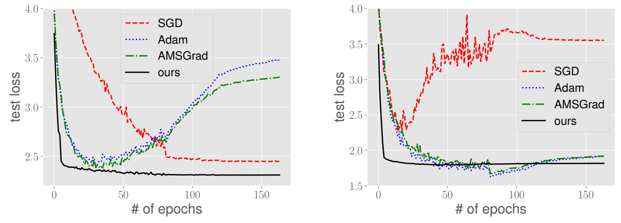
(a) AlexNet (b) VGG

Fig. 2. Test loss on Fashion-MNIST



(a) AlexNet (b) VGG

Fig. 3. Test loss on Cifar-10



(a) AlexNet (b) VGG

Fig. 4. Test loss on Cifar-100

we set the coefficient of momentum in Adam and AMSGrad to 0.99. The learning rates were 10^{-4} for SVHN/VGG, and 10^{-3} for the other combinations. In plain SGD, the learning rate was 10^{-1} for VGG and AlexNet/Fashion-MNIST. We set the learning rate to 10^{-2} for the other combinations. In our method, the learning rate was 10^{-2} for all settings. The switchover point was 70 for Fashion-MNIST/AlexNet, 25 for Fashion-MNIST/VGG, 20 for SVHN, 15 for Cifar-10/AlexNet, 10 for Cifar-10/100/VGG, and 5 for Cifar-100/AlexNet. All algorithms used the mini-batch size of 128. The number of training epochs was 164. The learning rate was divided by 10 for 81 and 122 epochs. We measured the test loss (negative log likelihood for test data) to investigate the generalization performance.

B. Results

Figures 1, 2, 3 and 4 show the test losses of each dataset and architecture. Our method reduced the test losses more rapidly than SGD, Adam and AMSGrad for all datasets and combinations. This is because our method utilizes block diagonal approximation for $F_\theta^{-1/2}$, and effectively updates parameters along a geodesic according to Theorem 1 and Corollary 1. On the other hand, since Adam and AMSGrad approximate $F_\theta^{-1/2}$ by using diagonal approximation, parameter update seldom follows a geodesic.

For many settings of Adam and AMSGrad, test losses increased in the last phase of training (Figure 1(b), Figure 2(b), Figure 3 and Figure 4). In more detail, test losses of Adam and AMSGrad in Figure 1(b) and 4(a) are larger than those of plain SGD, even though we tuned the hyper-parameters as described above. On the other hand, our method prevented any increase in test loss. Although the test loss of our method was temporarily larger than that of Adam and AMSGrad around 80 epochs in Figure 4(b), Adam and AMSGrad suffered increased test losses in the last phase of training, and our method finally outperformed both of them. These results suggest that the switching strategy based on Theorem 2 and Corollary 2 is effective in avoiding over fitting.

VI. Conclusion

We proposed TSO-ALRA, a two-stage optimizer using an adaptive learning rate algorithm, by analyzing the behaviors of existing adaptive learning rate algorithms when training DNNs. Our analysis examined two approaches suitable for DNNs: geodesics on the statistical manifold and covariance matrix of gradients. We revealed that existing adaptive learning rate algorithms have two properties: (i) they approximately update parameters along geodesics since they use $F_\theta^{-1/2}$, and (ii) the covariance matrix changes into an identity matrix around local minima as they use $F_\theta^{-1/2}$. Although property (i) achieves efficient training, standard adaptive learning rate algorithms such as Adam can not realize this benefit due to their use of diagonal approximation. To overcome this problem, we introduced a more accurate approximation technique than diagonal approximation such that parameter updates closely follow geodesics on the manifold. Property (ii) suggests that existing adaptive learning rate algorithms are likely to suffer degraded generalization performance because their diffusion around local minima is isotropic. Therefore, we switch to plain SGD in the last phase of training to prevent the covariance structure from breaking. Our image classification experiments on four well-known datasets showed that TSO-ALRA efficiently converges with high generalization performance, and validated our theoretical findings. Note that our theoretical findings are nontrivial because the role of $F_\theta^{-1/2}$ in adaptive learning rate algorithms has been unclear as regards DNN training whereas F_θ^{-1} in Natural Gradient is well understood.

References

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 1106–1114, 2012.

[2] Yasutoshi Ida and Yasuhiro Fujiwara. Network Implosion: Effective Model Compression for ResNets via Static Layer Pruning and Retraining. In *International Joint Conference on Neural Networks, IJCNN*, pages 1–8. IEEE, 2019.

[3] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-FCN: Object Detection via Region-based Fully Convolutional Networks. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 379–387, 2016.

[4] Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. Character-level Convolutional Networks for Text Classification. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 649–657, 2015.

[5] Andrew Gibiansky, Sercan Ömer Arik, Gregory Frederick Diamos, John Miller, Kainan Peng, Wei Ping, Jonathan Raiman, and Yanqi Zhou. Deep Voice 2: Multi-Speaker Neural Text-to-Speech. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 2966–2974, 2017.

[6] Herbert Robbins and Sutton Monro. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, pages 400–407, 1951.

[7] Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the Convergence of Adam and Beyond. In *International Conference on Learning Representations (ICLR)*, 2018.

[8] John Duchi, Elad Hazan, and Yoram Singer. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *The Journal of Machine Learning Research*, 12:2121–2159, 2011.

[9] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the Gradient by a Running Average of its Recent Magnitude. *COURSERA: Neural Networks for Machine Learning*, 2012.

[10] Matthew Zeiler. ADADELTA: An Adaptive Learning Rate Method. *arXiv preprint arXiv:1212.5701*, 2012.

[11] Diederik Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *International Conference in Learning Representations (ICLR)*, 2014.

[12] Yasutoshi Ida, Yasuhiro Fujiwara, and Sotetsu Iwamura. Adaptive Learning Rate via Covariance Matrix Based Preconditioning for Deep Neural Networks. In *IJCAI*, pages 1923–1929, 2017.

[13] Yann Dauphin, Harm de Vries, and Yoshua Bengio. Equilibrated Adaptive Learning Rates for Non-convex Optimization. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 1504–1512, 2015.

[14] Yann Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and Attacking the Saddle Point Problem in High-dimensional Non-convex Optimization. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 2933–2941, 2014.

[15] Zhanxing Zhu, Jingfeng Wu, Bing Yu, Lei Wu, and Jinwen Ma. The Anisotropic Noise in Stochastic Gradient Descent: Its Behavior of Escaping from Sharp Minima and Regularization Effects. In *ICML*, pages 7654–7663, 2019.

[16] Yasuhiro Fujiwara, Yasutoshi Ida, Hiroaki Shiokawa, and Sotetsu Iwamura. Fast Lasso Algorithm via Selective Coordinate Descent. In *AAAI*, pages 1561–1567, 2016.

[17] Yasutoshi Ida, Yasuhiro Fujiwara, and Hisashi Kashima. Fast Sparse Group Lasso. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 1700–1708, 2019.

[18] Shun-ichi Amari and Hiroshi Nagaoka. *Methods of Information Geometry*, volume 191. American Mathematical Soc., 2007.

[19] James Martens and Roger B. Grosse. Optimizing Neural Networks with Kronecker-factored Approximate Curvature. In *ICML*, pages 2408–2417, 2015.

[20] Stephan Mandt, Matthew D. Hoffman, and David M. Blei. Stochastic Gradient Descent as Approximate Bayesian Inference. *Journal of Machine Learning Research*, 18:134:1–134:35, 2017.

[21] T. P. Vogl, J. K. Mangis, A. K. Rigler, W. T. Zink, and D. L. Alkon. Accelerating the Convergence of the Back-propagation Method. *Biological Cybernetics*, 59(4):257–263, Sep 1988.

[22] Xiao-Hu Yu, Guo-An Chen, and Shixin Cheng. Dynamic Learning Rate Optimization of the Backpropagation Algorithm. *IEEE Trans. Neural Networks*, 6(3):669–677, 1995.

[23] Roberto Battiti. Accelerated Backpropagation Learning: Two Optimization Methods. *Complex Systems*, 3(4), 1989.

[24] Yann LeCun, Patrice Y. Simard, and Barak Pearlmutter. Automatic Learning Rate Maximization by On-Line Estimation of the Hessian's Eigenvectors. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 156–163. 1993.

[25] Robert A. Jacobs. Increased Rates of Convergence through Learning Rate Adaptation. *Neural Networks*, 1(4):295–307, 1988.

[26] Lu'is Almeida, Thibault Langlois, Jos'e D. Amaral, and Rua Alves Redol. On-Line Step Size Adaptation. Technical report, INESC. 9 Rua Alves Redol, 1000, 1997.

[27] Ashia C. Wilson, Rebecca Roelofs, Mitchell Stern, Nati Srebro, and Benjamin Recht. The Marginal Value of Adaptive Gradient Methods in Machine Learning. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 4151–4161, 2017.

[28] Shun-ichi Amari. Natural Gradient Works Efficiently in Learning. *Neural Computation*, 10(2):251–276, 1998.

[29] Roger B. Grosse and James Martens. A Kronecker-factored Approximate Fisher Matrix for Convolution Layers. In *ICML*, pages 573–582, 2016.

[30] Yuhuai Wu, Elman Mansimov, Roger B. Grosse, Shun Liao, and Jimmy Ba. Second-order Optimization for Deep Reinforcement Learning using Kronecker-factored Approximation. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 5285–5294, 2017.

[31] Lukas Balles and Philipp Hennig. Dissecting Adam: The Sign, Magnitude and Variance of Stochastic Gradients. In *ICML*, pages 413–422, 2018.

[32] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *CoRR*, abs/1609.08144, 2016.

[33] Nitish Shirish Keskar and Richard Socher. Improving Generalization Performance by Switching from Adam to SGD. *CoRR*, abs/1712.07628, 2017.

[34] Takuya Akiba, Shuji Suzuki, and Keisuke Fukuda. Extremely Large Minibatch SGD: Training ResNet-50 on ImageNet in 15 Minutes. *CoRR*, abs/1711.04325, 2017.

[35] Pierre Sermanet, Sandhya Chintala, and Yann LeCun. Convolutional Neural Networks Applied to House Numbers Digit Classification. In *International Conference on Pattern Recognition (ICPR)*, pages 3288–3291, 2012.

[36] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *CoRR*, abs/1708.07747, 2017.

[37] Alex Krizhevsky and Geoffrey Hinton. Learning Multiple Layers of Features from Tiny Images, 2009.

[38] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-scale Image Recognition. *arXiv preprint arXiv:1409.1556*, 2014.