Positive-Negative Momentum:
Manipulating Stochastic Gradient Noise to Improve Generalization

Zeke Xie [1]   **Li Yuan** [2]   **Zhanxing Zhu** [3]   **Masashi Sugiyama** [4 1]

Abstract

It is well-known that stochastic gradient noise (SGN) acts as implicit regularization for deep learning and is essentially important for both optimization and generalization of deep networks. Some works attempted to artificially simulate SGN by injecting random noise to improve deep learning. However, it turned out that the injected simple random noise cannot work as well as SGN, which is anisotropic and parameter-dependent. For simulating SGN at low computational costs and without changing the learning rate or batch size, we propose the Positive-Negative Momentum (PNM) approach that is a powerful alternative to conventional Momentum in classic optimizers. The introduced PNM method maintains two approximate independent momentum terms. Then, we can control the magnitude of SGN explicitly by adjusting the momentum difference. We theoretically prove the convergence guarantee and the generalization advantage of PNM over Stochastic Gradient Descent (SGD). By incorporating PNM into the two conventional optimizers, SGD with Momentum and Adam, our extensive experiments empirically verified the significant advantage of the PNM-based variants over the corresponding conventional Momentum-based optimizers. Code: `https://github.com/zeke-xie/Positive-Negative-Momentum`.

1. Introduction

Stochastic optimization methods, such as Stochastic Gradient Descent (SGD), have been popular and even necessary in training deep neural networks (LeCun et al., 2015).

It is well-known that stochastic gradient noise (SGN) in stochastic optimization acts as implicit regularization for deep learning and is essentially important for both optimization and generalization of deep neural networks (Hochreiter & Schmidhuber, 1995; 1997a; Hardt et al., 2016; Xie et al., 2021b; Wu et al., 2021). The theoretical mechanism behind the success of stochastic optimization, particularly the SGN, is still a fundamental issue in deep learning theory.

SGN matters. From the viewpoint of minima selection, SGN can help find flatter minima, which tend to yield better generalization performance (Hochreiter & Schmidhuber, 1995; 1997a). For this reason, how SGN selects flat minima has been investigated thoroughly. For example, Zhu et al. (2019) quantitatively revealed that SGN is better at selecting flat minima than trivial random noise, as SGN is anisotropic and parameter-dependent. Xie et al. (2021b) recently proved that, due to the anisotropic and parameter-dependent SGN, SGD favors flat minima even exponentially more than sharp minima. From the viewpoint of optimization dynamics, SGN can accelerate escaping from saddle points via stochastic perturbations to gradients (Jin et al., 2017; Daneshmand et al., 2018; Staib et al., 2019; Xie et al., 2020b).

Manipulating SGN. Due to the benefits of SGN, improving deep learning by manipulating SGN has become a popular topic. There are mainly two approaches along this line.

The first approach is to control SGN by tuning the hyperparameters, such as the learning rate and batch size, as the magnitude of SGN has been better understood recently. It is well-known (Mandt et al., 2017) that the magnitude of SGN in continuous-time dynamics of SGD is proportional to the ratio of the learning rate η and the batch size B, namely $\frac{\eta}{B}$. He et al. (2019) and Li et al. (2019) reported that increasing the ratio $\frac{\eta}{B}$ indeed can improve test performance due to the stronger implicit regularization of SGN. However, this method is limited and not practical for at least three reasons. First, training with a too small batch size is computationally expensive per epoch and often requires more epochs for convergence (Hoffer et al., 2017; Zhang et al., 2019). Second, increasing the learning rate only works in a narrow range, since too large initial learning rates may lead to optimization divergence or bad convergence (Keskar et al., 2017; Masters & Luschi, 2018; Xie et al., 2020a). In

---

[1]The University of Tokyo [2]National University of Singapore [3]Beijing Institute of Big Data Research [4]RIKEN Center for AIP. Correspondence to: Zeke Xie <xie@ms.k.u-tokyo.ac.jp>, Li Yuan <yuanli@u.nus.edu>, Zhanxing Zhu <zhanxing.zhu@pku.edu.cn>, Masashi Sugiyama <sugi@k.u-tokyo.ac.jp>.

practice, one definitely cannot use a too large learning rate to guarantee good generalization (Keskar et al., 2017; Masters & Luschi, 2018). Third, decaying the ratio $\frac{\eta}{B}$ during training (via learning rate decay) is almost necessary for the convergence of stochastic optimization as well as the training of deep networks (Smith et al., 2018). Thus, controlling SGN by adjusting the ratio $\frac{\eta}{B}$ cannot consistently be performed during the entire training process.

The second approach is to simulate SGN by using artificial noise. Obviously, we could simulate SGN well only if we clearly understood the structure of SGN. Related works (Daneshmand et al., 2018; Zhu et al., 2019; Xie et al., 2021b; Wen et al., 2020) studied the noise covariance structure, and there still exists a dispute about the noise type of SGD and its role in generalization (Simsekli et al., 2019; Xie et al., 2021b). It turned out that, while injecting small Gaussian noise into SGD may improve generalization (An, 1996; Neelakantan et al., 2015; Zhou et al., 2019; Xie et al., 2021a), unfortunately, Gradient Descent (GD) with artificial noise still performs much worse than SGD (i.e., SGD can be regarded as GD with SGN). Wu et al. (2020) argued that GD with multiplicative sampling noise may generalize as well as SGD, but the considered SGD baseline was weak due to the absence of weight decay and common training tricks. Thus, this approach cannot work well in practice.

Contribution. Is it possible to manipulate SGN without changing the learning rate or batch size? Yes. In this work, we propose Positive-Negative Momentum (PNM) for enhancing SGN at the low computational and coding costs. We summarize four contributions in the following.

- We introduce a novel method for manipulating SGN without changing the learning rate or batch size. The proposed PNM strategy can easily replace conventional Momentum in classical optimizers, including SGD and Adaptive Momentum Estimation (Adam).

- We theoretically prove that PNM has a convergence guarantee similar to conventional Momentum.

- Within the PAC-Bayesian framework, we theoretically prove that PNM may have a tighter generalization bound than SGD.

- We provide extensive experimental results to verify that PNM can indeed make significant improvements over conventional Momentum, shown in Table 1.

In Section 2, we introduce the proposed methods and the motivation of enhancing SGN. In Section 3, we present the convergence analysis. In Section 4, we present the generalization analysis. In Section 5, we conduct empirical analysis. In Section 6, we conclude our main work.

2. Methodology

In this section, we introduce the proposed PNM method and explain how it can manipulate SGN.

Notation. Suppose the loss function is $f(\theta)$, θ denotes the model parameters, the learning rate is η, the batch size is B, and the training data size is N. The basic gradient-descent-based updating rule can written as

$$\theta_{t+1} = \theta_t - \eta g_t. \tag{1}$$

Note that $g_t = \nabla f(\theta_t)$ for deterministic optimization, while $g_t = \nabla f(\theta_t) + \xi_t$ for stochastic optimization, where ξ_t indicates SGN. As SGN is from the difference between SGD and GD and the minibatch samples are uniformly chosen from the whole training dataset, it is commonly believed that g_t is an unbiased estimator of the true gradient $\nabla f(\theta_t)$ for stochastic optimization. Without loss of generality, we only consider one-dimensional case in Sections 2 and 3. The mean and the variance of SGN can be rewritten as $\mathbb{E}[\xi] = 0$ and $\mathrm{Var}(\xi) = \sigma^2$, respectively. We may use σ as a measure of the noise magnitude.

Conventional Momentum. We first introduce the conventional Momentum method, also called Heavy Ball (HB), seen in Algorithm 1 (Zavriev & Kostyuk, 1993). We obtain vanilla SGD by $\beta_1 = 0$ and $\beta_3 = 1$, and obtain common SGD with Momentum by $\beta_1 = 0.9$ and $\beta_3 = 1$. Algorithm 1 is the actual PyTorch SGD(Paszke et al., 2019), and can be theoretically reduced to SGD with a different learning rate. Adam uses the exponential moving average of past stochastic gradients as momentum by $\beta_3 = 1 - \beta_1$. The conventional Momentum can be written as

$$m_t = \sum_{k=0}^{t} \beta_3 \beta_1^{t-k} g_k, \tag{2}$$

which is the estimated gradient for updating model parameters. Then we approximately have $\mathbb{E}[m] \approx \frac{\beta_3}{1-\beta_1} \nabla f(\theta)$. The stochastic noise in momentum is given by

$$\xi_t^{\mathrm{hb}} = \sum_{k=0}^{t} \beta_3 \beta_1^{t-k} \xi_k. \tag{3}$$

Without loss of generality, we use the Adam-style Momentum with $\beta_3 = 1 - \beta_1$ in our analysis. Thus, the conventional momentum does not control the gradient noise magnitude, because, for large t,

$$\mathrm{Var}(\xi^{\mathrm{hb}}) = \beta_3 \frac{1 - \beta_1^{t+1}}{1 - \beta_1} \sigma^2 \approx \sigma^2. \tag{4}$$

We have assumed that SGN ξ is approximately independent. Since this approximation holds well in the limit of $\frac{B}{N} \to 0$, this assumption is common in theoretical analysis (Mandt et al., 2017).

Table 1. PNM versus conventional Momentum. We report the mean and the standard deviations (as the subscripts) of the optimal test errors computed over three runs of each experiment. The proposed PNM-based methods show significantly better generalization than conventional momentum-based methods. Particularly, as Theorem 4.3 indicates, Stochastic PNM indeed consistently outperforms the conventional baseline, SGD.

DATASET	MODEL	PNM	ADAPNM	SGD	ADAM	AMSGRAD	ADAMW	ADABOUND	PADAM	YOGI	RADAM
CIFAR-10	RESNET18	$\mathbf{4.48}_{0.09}$	$4.94_{0.05}$	$5.01_{0.03}$	$6.53_{0.03}$	$6.16_{0.18}$	$5.08_{0.07}$	$5.65_{0.08}$	$5.12_{0.04}$	$5.87_{0.12}$	$6.01_{0.10}$
	VGG16	$6.26_{0.05}$	$\mathbf{5.99}_{0.11}$	$6.42_{0.02}$	$7.31_{0.25}$	$7.14_{0.14}$	$6.48_{0.13}$	$6.76_{0.12}$	$6.15_{0.06}$	$6.90_{0.22}$	$6.56_{0.04}$
CIFAR-100	RESNET34	$20.59_{0.29}$	$\mathbf{20.41}_{0.18}$	$21.52_{0.37}$	$27.16_{0.55}$	$25.53_{0.19}$	$22.99_{0.40}$	$22.87_{0.13}$	$22.72_{0.10}$	$23.57_{0.12}$	$24.41_{0.40}$
	DENSENET121	$\mathbf{19.76}_{0.28}$	$20.68_{0.11}$	$19.81_{0.33}$	$25.11_{0.15}$	$24.43_{0.09}$	$21.55_{0.14}$	$22.69_{0.15}$	$21.10_{0.23}$	$22.15_{0.36}$	$22.27_{0.22}$
	GOOGLENET	$20.38_{0.31}$	$\mathbf{20.26}_{0.21}$	$21.21_{0.29}$	$26.12_{0.33}$	$25.53_{0.17}$	$21.29_{0.17}$	$23.18_{0.31}$	$21.82_{0.17}$	$24.24_{0.16}$	$22.23_{0.15}$

Basic Idea. Our basic idea for manipulating SGN is simple. Suppose that $g^{(a)}$ and $g^{(b)}$ are two independent unbiased estimators of $\nabla f(\theta)$. Then their weighted average is

$$\bar{g} = (1 + \beta_0)g^{(a)} - \beta_0 g^{(b)}$$
$$= \nabla f(\theta) + \bar{\xi}, \tag{5}$$

where $\bar{\xi} = (1 + \beta_0)\xi^{(a)} - \beta_0\xi^{(b)}$. If $\beta_0 > 0$, for the generated noisy gradient $\bar{g}$, we have $\mathbb{E}[\bar{g}] = \nabla f(\theta)$ and $\text{Var}(\bar{\xi}) = [(1 + \beta_0)^2 + \beta_0^2]\sigma^2$. In this way, we can control the noise magnitude by β_0 without changing the expectation of the noisy gradient.

Positive-Negative Momentum. Inspired by this simple idea, we combine the positive-negative averaging with the conventional Momentum method. For simplicity, we assume that t is an odd number. We maintain two independent momentum terms as

$$\begin{cases} m_t^{(\text{odd})} = \sum_{k=1,3,\ldots,t} \beta_3 \beta_1^{t-k} g_k, \\ m_t^{(\text{even})} = \sum_{k=0,2,\ldots,t-1} \beta_3 \beta_1^{t-k} g_k, \end{cases} \tag{6}$$

by using two alternate sequences of past gradients, respectively. Then we use the positive-negative average,

$$m_t = (1 + \beta_0)m_t^{(\text{odd})} - \beta_0 m_t^{(\text{even})}, \tag{7}$$

as an estimated gradient for updating model parameters. Similarly, if t is an even number, we let $m_t = (1 + \beta_0)m_t^{(\text{even})} - \beta_0 m_t^{(\text{odd})}$. When $\beta_0 > 0$, one momentum term has a positive coefficient and another one has a negative coefficient. Thus, we call it a positive-negative momentum pair.

The stochastic noise ξ_t^{pnm} in the positive-negative momentum pair is given by

$$\xi_t^{\text{pnm}} = (1 + \beta_0) \sum_{k=1,3,\ldots,t} \beta_3 \beta_1^{t-k} \xi_k - \beta_0 \sum_{k=0,2,\ldots,t-1} \beta_3 \beta_1^{t-k} \xi_k. \tag{8}$$

For large t, we write the noise variance as

$$\text{Var}(\xi^{\text{pnm}}) \approx [(1 + \beta_0)^2 + \beta_0^2]\sigma^2. \tag{9}$$

The noise magnitude of positive-negative momentum in Equation (7) is $\sqrt{(1 + \beta_0)^2 + \beta_0^2}$ times the noise magnitude of conventional Momentum.

While computing the gradients twice per iteration by using two minibatches can be also used for constructing positive-negative pairs, using past stochastic gradients has two benefits: lower computational costs and lower coding costs. First, avoiding computing the gradients twice per iteration save computation costs. Second, we may implement the proposed method inside optimizers, which can be employed in practice more easily.

Algorithm 1 (Stochastic) Heavy Ball/Momentum

$m_t = \beta_1 m_{t-1} + \beta_3 g_t$
$\theta_{t+1} = \theta_t - \eta m_t$

Algorithm 2 (Stochastic) PNM

$m_t = \beta_1^2 m_{t-2} + (1 - \beta_1^2)g_t$
$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{(1+\beta_0)^2+\beta_0^2}}[(1 + \beta_0)m_t - \beta_0 m_{t-1}]$

Algorithm 3 AdaPNM

$m_t = \beta_1^2 m_{t-2} + (1 - \beta_1^2)g_t$
$\hat{m}_t = \frac{(1+\beta_0)m_t - \beta_0 m_{t-1}}{(1-\beta_1^t)}$
$v_t = \beta_2 v_{t-1} + (1 - \beta_2)g_t^2$
$v_{\max} = \max(v_t, v_{\max})$
$\hat{v}_t = \frac{v_{\max}}{1-\beta_2^t}$
$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{(1+\beta_0)^2+\beta_0^2}(\sqrt{\hat{v}_t}+\epsilon)}\hat{m}_t$

Algorithms. We further incorporate PNM into SGD and Adam, and propose two novel PNM-based variants, including (Stochastic) PNM in Algorithm 2 and AdaPNM in Algorithm 3. Note that, by letting $\beta_0 = -\frac{\beta_1}{1+\beta_1}$, we may recover conventional Momentum and Adam as the special cases of Algorithm 2 and Algorithm 3. Note that our paper uses the AMSGrad variant in Algorithm 3 unless we specify it. Because Reddi et al. (2019) revealed that the AMSGrad variant to secure the convergence guarantee of adaptive gra-

dient methods. We supplied AdaPNM without AMSGrad in Appendix.

Normalizing the learning rate. We particularly normalize the learning rate by the noise magnitude as $\frac{\eta}{\sqrt{(1+\beta_0)^2+\beta_0^2}}$ in the proposed algorithms. The ratio of the noise magnitude to the learning rate matters to the convergence error (see Theorem 3.1 below). In practice (not the long-time limit), it is important to achieve low convergence errors in the same epochs as SGD. Practically, we also observe in experiments that using the learning rate as $\frac{\eta}{\sqrt{(1+\beta_0)^2+\beta_0^2}}$ for PNM can free us from re-tuning the hyperparameters, while we will need to re-fine-tune the learning rate without normalization, which is time-consuming in practice. Note that PNM can have a much larger ratio of the noise magnitude to learning rate than SGD.

3. Convergence Analysis

In this section, we theoretically prove the convergence guarantee of Stochastic PNM.

By Algorithm 2, we may rewrite the main updating rules as

$$
\begin{cases}
m_t = \beta_1^2 m_{t-2} + (1-\beta_1^2)g_t, \\
(\theta_{t+1} - \eta_0\beta_0 m_t) = (\theta_t - \eta_0\beta_0 m_{t-1}) - \eta_0 m_t,
\end{cases}
$$

where $\eta_0 = \frac{\eta}{\sqrt{(1+\beta_0)^2+\beta_0^2}}$. We denote that $\theta_{-2} = \theta_{-1} = \theta_0$, $\beta = \beta_1^2$, and $\alpha = \eta\frac{(1-\beta)}{\sqrt{(1+\beta_0)^2+\beta_0^2}}$. Then PNM can be written as

$$
\begin{cases}
x_t = \theta_t - \beta_0 m_{t-1}, \\
x_{t+1} = x_t - \alpha g_t + \beta(x_{t-1} - x_{t-2}).
\end{cases} \tag{10}
$$

Note that $x_{t+1} - x_t = \alpha m_t$. We may also write SGD with Momentum as

$$
\theta_{t+1} = \theta_t - \eta g_t + \beta(\theta_t - \theta_{t-1}). \tag{11}
$$

Obviously, PNM maintains two approximately independent momentum terms by using past odd-number-step gradients and even-number-step gradients, respectively.

Inspired by Yan et al. (2018), we propose Theorem 3.1 and prove that Stochastic PNM has a similar convergence rate to SGD with Momentum. The errors given by Stochastic PNM and SGD with Momentum are both $\mathcal{O}(\frac{1}{\sqrt{t}})$ (Yan et al., 2018). We leave all proofs in Appendix A.

Theorem 3.1 (Convergence of Stochastic Positive-Negative Momentum). *Assume that $f(\theta)$ is a L-smooth function, f is lower bounded as $f(\theta) \geq f^\star$, $\mathbb{E}[\xi] = 0$, $\mathbb{E}[\|g(\theta,\xi) - \nabla f(\theta)\|^2] \leq \sigma^2$, and $\|\nabla f(\theta)\| \leq G$ for any θ. Let $\beta_1 \in [0,1)$, $\beta_0 \geq 0$ and Stochastic PNM run for $t+1$ iterations.*

If $\frac{\eta}{\sqrt{(1+\beta_0)^2+\beta_0^2}} = \min\{\frac{1}{2L}, \frac{C}{\sqrt{t+1}}\}$, we have

$$
\min_{k=0,\dots,t} \mathbb{E}[\|\nabla f(\theta_k)\|^2]
$$
$$
\leq \frac{2(f(\theta_0) - f^\star)}{t+1}\max\{2L, \frac{\sqrt{t+1}}{C}\} + \frac{C_1}{\sqrt{t+1}},
$$

where

$$
C_1 = C\frac{L(\beta + \beta_0(1-\beta))^2(G^2 + \sigma^2) + L(1-\beta)^2\sigma^2}{(1-\beta)^2}.
$$

4. Generalization Analysis

In this section, we theoretically prove the generalization advantage of Stochastic PNM over SGD by using PAC-Bayesian framework (McAllester, 1999b;a). We discover that, due to the stronger SGN, the posterior given by Stochastic PNM has a tighter generalization bound than the SGD posterior.

4.1. The posterior analysis

McAllester (1999b) pointed that the posterior given by a training algorithm is closely related to the generalization bound. Thus, we first analyze the posteriors given by SGD, Momentum, and Stochastic Positive-Negative Momentum. Mandt et al. (2017) studied the posterior given by the continuous-time dynamics of SGD. We present Assumptions 1 in Mandt et al. (2017), which is true near minima. Note that $H(\theta)$ denotes the Hessian of the loss function f at θ.

Assumption 1 (The second-order Taylor approximation). *The loss function around a minimum $\theta^\star$ can be approximately written as*

$$
f(\theta) = f(\theta^\star) + \frac{1}{2}(\theta - \theta^\star)^\top H(\theta^\star)(\theta - \theta^\star).
$$

When the noisy gradient for each iteration is the unbiased estimator of the true gradient, the dynamics of gradient-based optimization can be always written as

$$
\theta_{t+1} = \theta_t - \eta(\nabla f(\theta) + C(\theta)\xi), \tag{12}
$$

where $C(\theta)$ is the covariance of gradient noise and ξ obeys the standard Gaussian distribution $\mathcal{N}(0, I)$. While recent papers (Simsekli et al., 2019; Xie et al., 2021b) argued about the noise types, we follow most papers (Welling & Teh, 2011; Jastrzkebski et al., 2017; Li et al., 2017; Mandt et al., 2017; Hu et al., 2019; Xie et al., 2021b) in this line and still approximate SGN as Gaussian noise due to Central Limit Theorem. The corresponding continuous-time dynamics can be written as

$$
d\theta = -\nabla f(\theta)dt + [\eta C(\theta)]^{\frac{1}{2}}dW_t, \tag{13}
$$

where $dW_t = \mathcal{N}(0, Idt)$ is a Wiener process.

Mandt et al. (2017) used SGD as an example and demonstrated that, the posterior generated by Equation (13) in the local region around a minimum $\theta^\star$ is a Gaussian distribution $\mathcal{N}(\theta^\star, \Sigma_{sgd})$. This well-known result can be formulated as Theorem 4.1. We denote that $H(\theta^\star) = H$ and $C(\theta^\star) = C$ in the following analysis.

Theorem 4.1 (The posterior generated by Equation (13) near a minimum (Mandt et al., 2017)). *Suppose that Assumption 1 holds, the covariance near the minimum $\theta^\star$ is $C(\theta^\star)$, the dynamics is governed by Equation (13). Then, in the long-time limit, the generated posterior Q near $\theta^\star$ is a Gaussian distribution $\mathcal{N}(\theta^\star, \Sigma)$. and the Σ satisfies*

$$\Sigma H(\theta^\star) + H(\theta^\star)\Sigma = \eta C(\theta^\star).$$

The vanilla SGD posterior. In the case of SGD, based on Jastrzebski et al. (2017); Zhu et al. (2019); Xie et al. (2021b), the covariance $C(\theta)$ is proportional to the Hessian $H(\theta)$ and inverse to the batch size B near minima:

$$C_{sgd}(\theta) \approx \frac{1}{B}H(\theta). \tag{14}$$

Equation (14) has been theoretically and empirically studied by related papers (Xie et al., 2021b; 2020b). Please see Appendix D for the details.

By Equation (14) and Theorem 4.1, we may further express Σ_{sgd} as

$$\Sigma_{sgd} = \frac{\eta}{2B}I, \tag{15}$$

where I is the $n \times n$ identity matrix and n is the number of model parameters.

The Momentum posterior. By Equation (3), we know that, in continuous-time dynamics, we may write the noise covariance in HB/SGD with Momentum as

$$C_{hb}(\theta) = \frac{\beta_3}{1 - \beta_1}C_{sgd} = C_{sgd}. \tag{16}$$

Without loss of generality, we have assumed that $\beta_3 = 1 - \beta_1$ in HB/SGD with Momentum. Then, in the long-time limit, we further express Σ_{hb} as

$$\Sigma_{hb} = \frac{\beta_3}{1 - \beta_1}\Sigma_{sgd} = \frac{\eta}{2B}I. \tag{17}$$

This result has also been obtained by Mandt et al. (2017). Thus, the Momentum posterior is approximately equivalent to the vanilla SGD posterior. In the PAC-Bayesian framework (Theorem 4.2), HB should generalize as well as SGD in the long-time limit.

The Stochastic PNM posterior. Similarly, by Equation (8), we know that, in continuous-time dynamics, we may write the noise covariance in Stochastic PNM as

$$C_{pnm}(\theta) = [(1 + \beta_0)^2 + \beta_0^2]C_{sgd}. \tag{18}$$

where $C_{pnm}(\theta) = ((1+\beta_0)^2+\beta_0^2)C_{sgd}(\theta)$ is the covariance of SGN in Stochastic PNM.

By Equation (14) and Theorem 4.1, in the long-time limit, we may further express Σ_{pnm} as

$$\Sigma_{pnm} = [(1 + \beta_0)^2 + \beta_0^2]\frac{\eta}{2B}I. \tag{19}$$

Thus, we may use the hyperparameter β_0 to rescale the covariance of the posterior. In the following analysis, we will prove that the PAC-Bayesian bound may heavily depend on the new hyperparameter β_0.

4.2. The PAC-Bayesian bound analysis

The PAC-Bayesian generalization bound. The PAC-Bayesian framework provides guarantees on the expected risk of a randomized predictor (hypothesis) that depends on the training dataset. The hypothesis is drawn from a distribution Q and sometimes referred to as a posterior given a training algorithm. We then denote the expected risk with respect to the distribution Q as $R(Q)$ and the empirical risk with respect to the distribution Q as $\hat{R}(Q)$. Note that P is typically assumed to be a Gaussian prior, $\mathcal{N}(0, \lambda^{-1}I)$, over the weight space Θ, where λ is the L_2 regularization strength (Graves, 2011; Neyshabur et al., 2017; He et al., 2019).

Assumption 2. *The prior over model weights is Gaussian, $P = \mathcal{N}(0, \lambda^{-1}I)$.*

We introduce the classical PAC-Bayesian generalization bound in Theorem 4.2.

Theorem 4.2 (The PAC-Bayesian Generalization Bound (McAllester, 1999b)). *For any real $\Delta \in (0, 1)$, with probability at least $1 - \Delta$, over the draw of the training dataset S, the expected risk for all distributions Q satisfies*

$$R(Q) - \hat{R}(Q) \leq 4\sqrt{\frac{1}{N}[KL(Q\|P) + \ln(\frac{2N}{\Delta})]},$$

where $KL(Q\|P)$ denotes the Kullback–Leibler divergence from P to Q.

We define that $Gen(Q) = R(Q) - \hat{R}(Q)$ is the expected generalization gap. Then the upper bound of $Gen(Q)$ can be written as

$$Sup\, Gen(Q) = 4\sqrt{\frac{1}{N}[KL(Q\|P) + \ln(\frac{2N}{\Delta})]}. \tag{20}$$

Theorem 4.2 demonstrates that the expected generalization gap's upper bound closely depends on the posterior given by a training algorithm. Thus, it is possible to improve generalization by decreasing $\mathrm{KL}(Q\|P)$.

The Kullback–Leibler divergence from P to Q. Note that Q and P are two Gaussian distributions, $\mathcal{N}(\mu_Q, \Sigma_Q)$ and $\mathcal{N}(\mu_P, \Sigma_P)$, respectively. Then we have

$$\mathrm{KL}(Q\|P) = \frac{1}{2}\left[\log\frac{\det(\Sigma_P)}{\det(\Sigma_Q)} + \mathrm{Tr}(\Sigma_P^{-1}\Sigma_Q)\right] + \frac{1}{2}(\mu_Q - \mu_P)^\top \Sigma_P^{-1}(\mu_Q - \mu_P) - \frac{n}{2}. \quad (21)$$

Suppose that $\mu_Q = \theta^\star$ and $\Sigma_Q(\gamma) = \gamma\Sigma_{sgd}$ correspond to the covariance-rescaled SGD posterior, and $\mu_P = 0$ and $\Sigma_P = \lambda^{-1}I$ correspond to L_2 regularization. Here $\gamma = ((1+\beta_0)^2 + \beta_0^2) \geq 1$. Then $\mathrm{KL}(Q(\gamma)\|P)$ is a function of γ, written as

$$\mathrm{KL}(Q(\gamma)\|P) = \frac{1}{2}\log\frac{\lambda^{-n}}{\gamma^n \det(\Sigma_{sgd})} + \frac{1}{2}\lambda^{-1}\gamma\,\mathrm{Tr}(\Sigma_{sgd}) + \frac{\lambda}{2}\|\theta^\star\|^2 - \frac{n}{2}. \quad (22)$$

For minimizing $\mathrm{KL}(Q(\gamma)\|P)$, we calculate its gradient with respect to γ as

$$\nabla_\gamma \mathrm{KL}(Q(\gamma)\|P) = \frac{n}{2}\left(\frac{\eta}{2B\lambda} - \frac{1}{\gamma}\right), \quad (23)$$

where we have used Equation (15).

It shows that $\mathrm{KL}(Q(\gamma)\|P)$ is a monotonically decreasing function on the interval of $\gamma \in [1, \frac{2B\lambda}{\eta}]$. Obviously, when $\frac{\eta}{2B\lambda} < 1$ holds, we can always decrease $\mathrm{KL}(Q(\gamma)\|P)$ by fine-tuning $\gamma > 1$. Note that Momentum is a special case of PNM with $\beta_0 = -\frac{\beta_1}{1+\beta_1}$, and the Momentum posterior is approximately equivalent to the vanilla SGD posterior.

Stochastic PNM may have a tighter bound than SGD. Based on the results above, we naturally prove Theorem 4.3 that Stochastic PNM can always have a tighter upper bound of the generalization gap than SGD by fine-tuning $\beta_0 > 0$ in any task where $\frac{\eta}{2B\lambda} < 1$ holds for SGD. In principle, SHB may be reduced to SGD with a different learning rate. SGD in PyTorch is actually equivalent to SHB. Thus, our theoretical analysis can be easily generalized to SHB.

Theorem 4.3 (The generalization advantage of Stochastic PNM). *Suppose Assumption 2, the conditions of Theorem 4.1, and Theorem 4.2 hold. If $\frac{\eta}{2B\lambda} < 1$ holds for SGD, then there must exist $\beta_0 \in (0, \frac{2B\lambda}{\eta}]$ that makes the following hold in the long-time limit:*

$$\mathrm{Sup}\,\mathrm{Gen}(Q_{\mathrm{pnm}}) < \mathrm{Sup}\,\mathrm{Gen}(Q_{sgd}).$$

When does $\frac{\eta}{2B\lambda} < 1$ hold? We do not theoretically prove that $\frac{\eta}{2B\lambda} < 1$ always holds. However, it is very easy to

empirically verify the inequality $\frac{\eta}{2B\lambda} < 1$ in any specific practical task. Fortunately, we find that $\frac{\eta}{2B\lambda} < 1$ holds in wide practical applications. For example, we have $\frac{\eta}{2B\lambda} \approx 0.039$ for the common setting that $\eta = 0.001$ (after the final learning rate decay), $B = 128$, and $\lambda = 1e-4$. It means that the proposed PNM method may improve generalization in wide applications.

How to select β_0 in practice? Recent work (He et al., 2019) suggested that increasing $\frac{\eta}{B}$ always improves generalization by using the PAC-Bayesian framework similarly to our work. However, as we discussed in Section 1, this is not true in practice for multiple reasons.

Similarly, while Equation (23) suggests that $\gamma = \frac{2B\lambda}{\eta}$ can minimize $\mathrm{Sup}\,\mathrm{Gen}(Q_{\mathrm{pnm}})$ in principle, we do not think that $\gamma = \frac{2B\lambda}{\eta}$ should always be the optimal setting in practice. Instead, we suggest that a γ slightly larger than one is good enough in practice. In our paper, we choose $\gamma = 5$ as the default setting, which corresponds to $\beta_0 = 1$.

We do not choose $\gamma = \frac{2B\lambda}{\eta}$ mainly because a too large γ requires too many iterations to reach the long-time limit. Theorem 3.1 also demonstrates that it will require much more iterations to reach convergence if β_0 is too large. However, in practice, the number of training epochs is usually fixed and is not consider as a fine-tuned hyperparameter. This supports our belief that PNM with γ slighter larger than one can be a good and robust default setting without re-tuning the hyperparameters.

5. Empirical Analysis

In this section, we empirically study how the PNM-based optimizers are compared with conventional optimizers.

Models and Datasets. We trained popular deep models, including ResNet18/ResNet34/ResNet50 (He et al., 2016), VGG16 (Simonyan & Zisserman, 2014), DenseNet121 (Huang et al., 2017), GoogLeNet (Szegedy et al., 2015), and Long Short-Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997b) on CIFAR-10/CIFAR-100 (Krizhevsky & Hinton, 2009), ImageNet (Deng et al., 2009) and Penn Tree-Bank (Marcus et al., 1993). We leave the implementation details in Appendix B.

Image Classification on CIFAR-10 and CIFAF-100. In Table 1, we first empirically compare PNM and AdaPNM with popular stochastic optimizers, including SGD, Adam (Kingma & Ba, 2015), AMSGrad (Reddi et al., 2019), AdamW (Loshchilov & Hutter, 2018), AdaBound (Luo et al., 2019), Padam (Chen & Gu, 2018), Yogi (Zaheer et al., 2018), and RAdam (Liu et al., 2019) on CIFAR-10 and CIFAR-100. It demonstrates that PNM-based optimizers generalize significantly better than the corresponding conventional Momentum-based optimizers. In Figure 1,
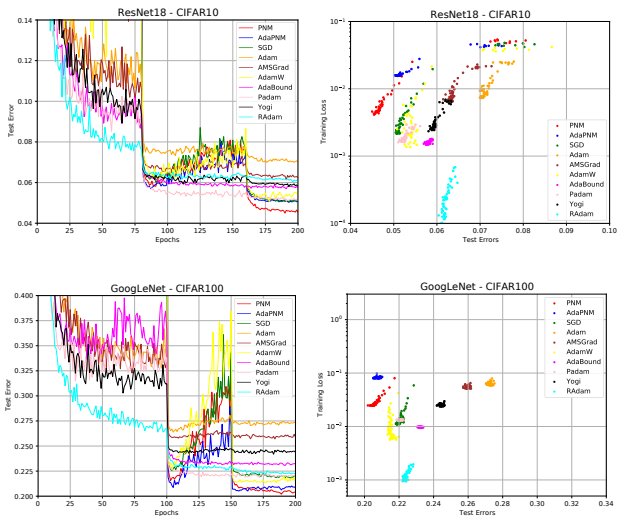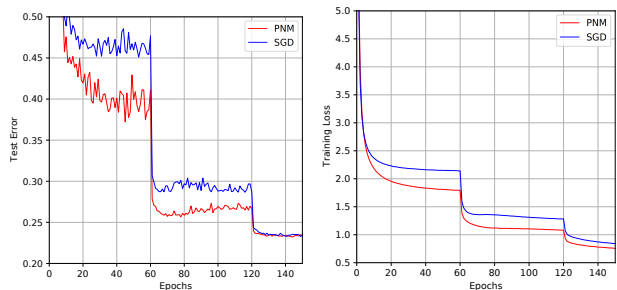
Figure 1. The learning curves of popular models on CIFAR-10 and CIFAR-100, respectively. Left Column: Test curves. Right Column: The scatter plots of training losses and test errors during final 40 epochs. It demonstrates that PNM and AdaPNM yield significantly better test results.

Table 2. Top-1 and top-5 test errors of ResNet50 on ImageNet. Note that the popular SGD baseline performace of ResNet50 on ImageNet has the test errors as 23.85% in PyTorch and 24.9% in He et al. (2016), which are both worse than our SGD baseline. AdaPNM (with decoupled weight decay and no amsgrad) significantly outperforms its conventional variant, Adam (with decoupled weight decay and no amsgrad).

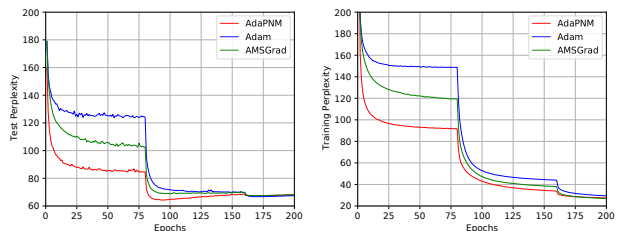|	PNM	SGD	ADAPNM	ADAMW
TOP1	**23.21**	23.28	**23.12**	23.62
TOP5	**6.73**	6.75	**6.82**	7.09

Figure 3. The learning curves of LSTM on Penn Treebank. The optimal test perplexity of AdaPNM, Adam, and AMSGrad are 64.25, 66.67, and 67.40, respectively. AdaPNM not only converges much faster than Adam and AMSGrad, but also yields lower test perplexity.
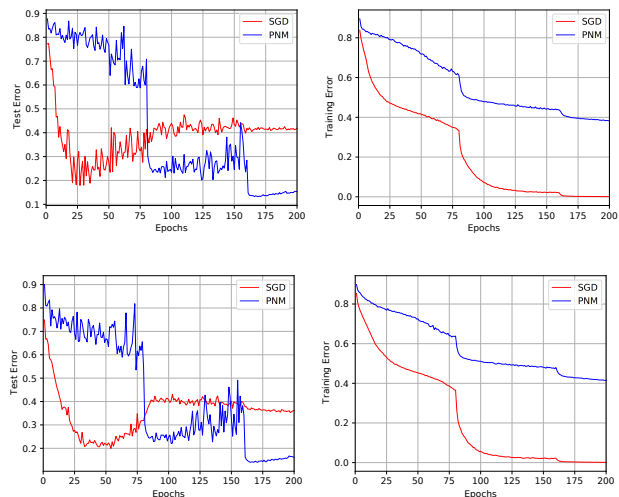

	Epoch60	Epoch120	Epoch150
PNM(Top1)	**37.23**	**25.67**	**23.21**
SGD(Top1)	45.10	28.66	23.28
PNM(Top5)	**14.53**	**7.79**	**6.73**
SGD(Top5)	19.04	9.30	6.75

Figure 2. The learning curves of ResNet50 on ImageNet. Left Subfigure: Top 1 Test Error. Right Subfigure: Training Loss. PNM not only generalizes significantly better than conventional momentum, but also converges faster. PNM always achieves lower training losses and test errors at the final epoch of each learning rate decay phase.

Figure 4. We compare PNM and SGD (with Momentum) by training ResNet34 on CIFAR-10 with 40% asymmetric label noise (Top Row) and 40% symmetric label noise (Bottom Row). Left: Test Curve. Right: Training Curve. We observe that PNM with a large β_0 may effectively relieve memorizing noisy labels and almost only learn clean labels, while SGD almost memorize all noisy labels and has a nearly zero training error.
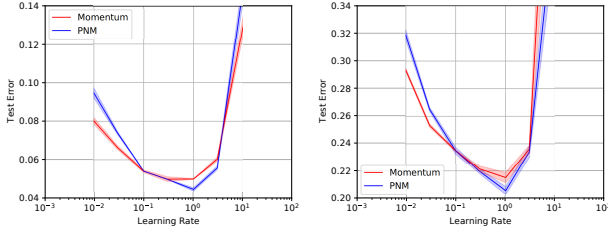
Figure 5. We compare the generalization of PNM and SGD under various learning rates. Left: ResNet18 on CIFAR-10. Right: ResNet34 on CIFAR-100. The optimal performance of PNM is better than the conventional SGD.

we note that PNM-based optimizers have better test performance even with similar training losses.

Image Classification on ImageNet. Table 2 also supports that the PNM-based optimizers generalizes better than the corresponding conventional Momentum-based optimizers. Figure 2 shows that, on the experiment on ImageNet, Stochastic PNM consistently has lower test errors than SGD at the final epoch of each learning rate decay phase. It indicates that PNM not only generalizes better, but also converges faster on ImageNet due to stronger SGN.

Language Modeling. As Adam is the most popular optimizer on Natural Language Processing tasks, we further compare AdaPNM (without amsgrad) with Adam (without amsgrad) as the baseline on the Language Modeling experiment. Figure 3 shows that AdaPNM outperforms the conventional Adam in terms of test performance and training speed.

Learning with noisy labels. Deep networks can easily overfit training data even with random labels (Zhang et al., 2017). We run PNM and SGD on CIFAR-10 with 40% label noise for comparing the robustness to noise memorization. Figure 4 shows that PNM has much better generalization than SGD and outperforms SGD by more 20 points at the final epoch. It also demonstrates that enhancing SGN may effectively mitigate overfitting training data.

Robustness to the learning rate and weight decay. In Figure 5, we show that PNM can consistently outperform SGD under a wide range of learning rates. Figure 6 further supports that PNM can be more robust to learning rates and weight decay than SGD, because PNM has a significantly deeper and wider basin in terms of test errors. This makes PNM a robust alternative to conventional Momentum.

Robustness to the new hyperparameter β_0. Finally, we empirically study how PNM depends on the hyperparameter β_0 in practice in Figure 7. The result in Figure 7 fully supports our motivation and theoretical analysis. It demonstrates that PNM may achieve significantly better general-
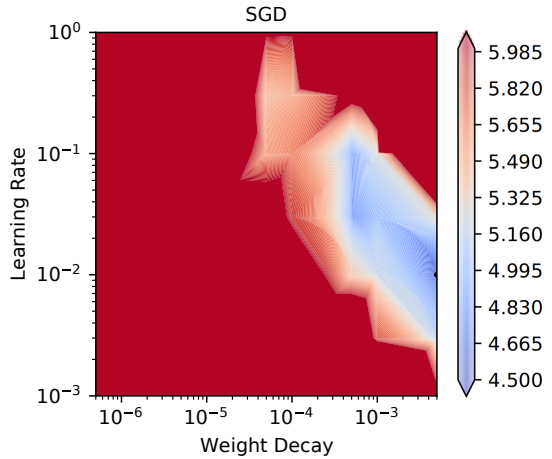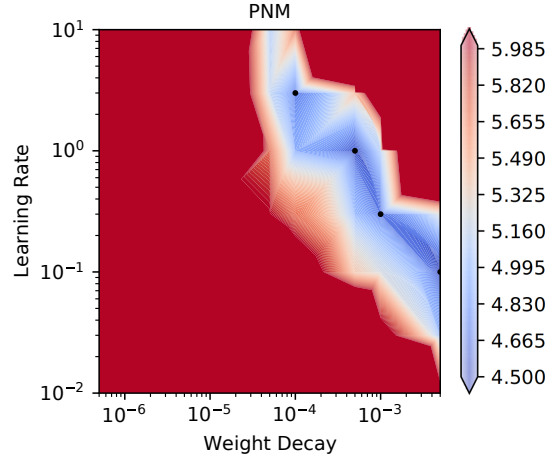

Figure 6. The test errors of ResNet18 on CIFAR-10 under various learning rates and weight decay. PNM has a much deeper and wider blue region near dark points ($\leq 4.83\%$) than SGD.
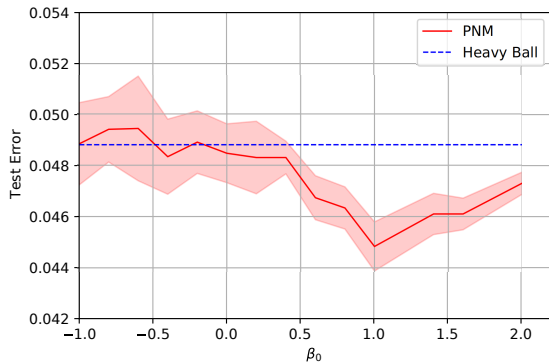

Figure 7. We train ResNet18 on CIFAR-10 under various β_0 choices. It demonstrates that PNM may achieve significantly better generalization with a wide range of $\beta_0 > 0$, which corresponds to a positive-negative momentum pair for enhancing SGN as we expect. With any $\beta_0 \in [-1, 0]$, the test performance does not sensitively depend on β_0, because this case cannot enhance SGN.

ization by choosing a proper $\beta_0 > 0$, which corresponds to a positive-negative momentum pair for enhancing SGN as we expect. With any $\beta_0 \in [-1, 0]$, the test performance does not sensitively depend on β_0. Because the case that $\beta_0 \in [-1, 0]$ corresponds to a positive-positive momentum pair and, thus, cannot enhance SGN.

Supplementary experiments. Please refer to Appendix C.

6. Conclusion

We propose a novel Positive-Negative Momentum method for manipulating SGN by using the difference of past gradients. The simple yet effective method can provably improve deep learning at very low costs. In practice, the PNM method is a powerful and robust alternative to the conventional Momentum method in classical optimizers and can usually make significant improvements.

While we only use SGD and Adam as the conventional base optimizers, it is easy to incorporate PNM into other advanced optimizers. Considering the importance and the popularity of Momentum, we believe that the proposed PNM indicates a novel and promising approach to designing optimization dynamics by manipulating gradient noise.

Acknowledgement

References

An, G. The effects of adding noise during backpropagation training on a generalization performance. *Neural computation*, 8(3):643–674, 1996.

Chen, J. and Gu, Q. Closing the generalization gap of adaptive gradient methods in training deep neural networks. *arXiv preprint arXiv:1806.06763*, 2018.

Daneshmand, H., Kohler, J., Lucchi, A., and Hofmann, T. Escaping saddles with stochastic gradients. In *International Conference on Machine Learning*, pp. 1155–1164, 2018.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Graves, A. Practical variational inference for neural networks. In *Advances in neural information processing systems*, pp. 2348–2356, 2011.

Hardt, M., Recht, B., and Singer, Y. Train faster, generalize better: Stability of stochastic gradient descent. In *International Conference on Machine Learning*, pp. 1225–1234, 2016.

He, F., Liu, T., and Tao, D. Control batch size and learning rate to generalize well: Theoretical and empirical evidence. In *Advances in Neural Information Processing Systems*, pp. 1141–1150, 2019.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hochreiter, S. and Schmidhuber, J. Simplifying neural nets by discovering flat minima. In *Advances in neural information processing systems*, pp. 529–536, 1995.

Hochreiter, S. and Schmidhuber, J. Flat minima. *Neural Computation*, 9(1):1–42, 1997a.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997b.

Hoffer, E., Hubara, I., and Soudry, D. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. In *Advances in Neural Information Processing Systems*, pp. 1729–1739, 2017.

Hu, W., Li, C. J., Li, L., and Liu, J.-G. On the diffusion approximation of nonconvex stochastic gradient descent. *Annals of Mathematical Sciences and Applications*, 4(1): 3–32, 2019.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

Jastrzkebski, S., Kenton, Z., Arpit, D., Ballas, N., Fischer, A., Bengio, Y., and Storkey, A. Three factors influencing minima in sgd. *arXiv preprint arXiv:1711.04623*, 2017.

Jin, C., Ge, R., Netrapalli, P., Kakade, S. M., and Jordan, M. I. How to escape saddle points efficiently. In *International Conference on Machine Learning*, pp. 1724–1732. PMLR, 2017.

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. On large-batch training for deep learning: Generalization gap and sharp minima. In *International Conference on Learning Representations*, 2017.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *3rd International Conference on Learning Representations, ICLR 2015*, 2015.

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. 2009.

LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *nature*, 521(7553):436, 2015.

Li, Q., Tai, C., et al. Stochastic modified equations and adaptive stochastic gradient algorithms. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 2101–2110. JMLR. org, 2017.

Li, Y., Wei, C., and Ma, T. Towards explaining the regularization effect of initial large learning rate in training neural networks. In *Advances in Neural Information Processing Systems*, pp. 11669–11680, 2019.

Liu, L., Jiang, H., He, P., Chen, W., Liu, X., Gao, J., and Han, J. On the variance of the adaptive learning rate and beyond. In *International Conference on Learning Representations*, 2019.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2018.

Luo, L., Xiong, Y., Liu, Y., and Sun, X. Adaptive gradient methods with dynamic bound of learning rate. *7th International Conference on Learning Representations, ICLR 2019*, 2019.

Mandt, S., Hoffman, M. D., and Blei, D. M. Stochastic gradient descent as approximate bayesian inference. *The Journal of Machine Learning Research*, 18(1):4873–4907, 2017.

Marcus, M., Santorini, B., and Marcinkiewicz, M. A. Building a large annotated corpus of english: The penn treebank. 1993.

Masters, D. and Luschi, C. Revisiting small batch training for deep neural networks. *arXiv preprint arXiv:1804.07612*, 2018.

McAllester, D. A. Pac-bayesian model averaging. In *Proceedings of the twelfth annual conference on Computational learning theory*, pp. 164–170, 1999a.

McAllester, D. A. Some pac-bayesian theorems. *Machine Learning*, 37(3):355–363, 1999b.

Neelakantan, A., Vilnis, L., Le, Q. V., Sutskever, I., Kaiser, L., Kurach, K., and Martens, J. Adding gradient noise improves learning for very deep networks. *arXiv preprint arXiv:1511.06807*, 2015.

Neyshabur, B., Bhojanapalli, S., McAllester, D., and Srebro, N. Exploring generalization in deep learning. In *Advances in Neural Information Processing Systems*, pp. 5947–5956, 2017.

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

Reddi, S. J., Kale, S., and Kumar, S. On the convergence of adam and beyond. *6th International Conference on Learning Representations, ICLR 2018*, 2019.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Simsekli, U., Sagun, L., and Gurbuzbalaban, M. A tail-index analysis of stochastic gradient noise in deep neural networks. In *International Conference on Machine Learning*, pp. 5827–5837, 2019.

Smith, S. L., Kindermans, P.-J., and Le, Q. V. Don't decay the learning rate, increase the batch size. In *International Conference on Learning Representations*, 2018.

Staib, M., Reddi, S., Kale, S., Kumar, S., and Sra, S. Escaping saddle points with adaptive gradient methods. In *International Conference on Machine Learning*, pp. 5956–5965. PMLR, 2019.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

Welling, M. and Teh, Y. W. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 681–688, 2011.

Wen, Y., Luk, K., Gazeau, M., Zhang, G., Chan, H., and Ba, J. An empirical study of stochastic gradient descent with structured covariance noise. In Chiappa, S. and Calandra, R. (eds.), *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pp. 3621–3631. PMLR, 26–28 Aug 2020.

Wu, J., Hu, W., Xiong, H., Huan, J., Braverman, V., and Zhu, Z. On the noisy gradient descent that generalizes as sgd. In *International Conference on Machine Learning*, pp. 10367–10376. PMLR, 2020.

Wu, J., Zou, D., Braverman, V., and Gu, Q. Direction matters: On the implicit regularization effect of stochastic gradient descent with moderate learning rate. *International Conference on Learning Representations*, 2021.

Xie, Z., Sato, I., and Sugiyama, M. Stable weight decay regularization. *arXiv preprint arXiv:2011.11152*, 2020a.

Xie, Z., Wang, X., Zhang, H., Sato, I., and Sugiyama, M. Adai: Separating the effects of adaptive learning rate and momentum inertia. *arXiv preprint arXiv:2006.15815*, 2020b.

Xie, Z., He, F., Fu, S., Sato, I., Tao, D., and Sugiyama, M. Artificial neural variability for deep learning: On overfitting, noise memorization, and catastrophic forgetting. *Neural Computation*, 2021a.

Xie, Z., Sato, I., and Sugiyama, M. A diffusion theory for deep learning dynamics: Stochastic gradient descent exponentially favors flat minima. In *International Conference on Learning Representations*, 2021b.

Yan, Y., Yang, T., Li, Z., Lin, Q., and Yang, Y. A unified analysis of stochastic momentum methods for deep learning. In *IJCAI International Joint Conference on Artificial Intelligence*, 2018.

Zaheer, M., Reddi, S., Sachan, D., Kale, S., and Kumar, S. Adaptive methods for nonconvex optimization. In *Advances in neural information processing systems*, pp. 9793–9803, 2018.

Zavriev, S. and Kostyuk, F. Heavy-ball method in nonconvex optimization problems. *Computational Mathematics and Modeling*, 4(4):336–341, 1993.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. Understanding deep learning requires rethinking generalization. In *International Conference on Machine Learning*, 2017.

Zhang, G., Li, L., Nado, Z., Martens, J., Sachdeva, S., Dahl, G., Shallue, C., and Grosse, R. B. Which algorithmic choices matter at which batch sizes? insights from a noisy quadratic model. In *Advances in Neural Information Processing Systems*, pp. 8196–8207, 2019.

Zhou, M., Liu, T., Li, Y., Lin, D., Zhou, E., and Zhao, T. Toward understanding the importance of noise in training neural networks. In *International Conference on Machine Learning*, 2019.

Zhu, Z., Wu, J., Yu, B., Wu, L., and Ma, J. The anisotropic noise in stochastic gradient descent: Its behavior of escaping from sharp minima and regularization effects. In *ICML*, pp. 7654–7663, 2019.