APOLLO: AN ADAPTIVE PARAMETER-WISE DIAGONAL QUASI-NEWTON METHOD FOR NONCONVEX STOCHASTIC OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

In this paper, we introduce APOLLO, a quasi-Newton method for nonconvex stochastic optimization, which dynamically incorporates the curvature of the loss function by approximating the Hessian via a diagonal matrix. Importantly, the update and storage of the diagonal approximation of Hessian is as efficient as adaptive first-order optimization methods with linear complexity for both time and memory. To handle nonconvexity, we replace the Hessian with its rectified absolute value, which is guaranteed to be positive-definite. Experiments on three tasks of vision and language show that APOLLO achieves significant improvements over other stochastic optimization methods, including SGD and variants of Adam, in terms of both convergence speed and generalization performance. The implementation of the algorithm is available at *anonymous link*.

1 INTRODUCTION

Nonconvex stochastic optimization is of core practical importance in many fields of machine learning, in particular for training deep neural networks (DNNs). First-order gradient-based optimization algorithms, conceptually attractive due to their linear efficiency on both the time and memory complexity, have led to tremendous progress and impressive successes. A number of advanced first-order algorithms have emerged over the years to pursue fast and stable convergence, among which stochastic gradient descent (SGD) (Robbins & Monro, 1951; LeCun et al., 1998), equipped with momentum (Rumelhart et al., 1985; Qian, 1999; Bottou & Bousquet, 2008), has stood out for its simplicity and effectiveness across a wide range of applications (Hinton & Salakhutdinov, 2006; Hinton et al., 2012; Graves, 2013). However, one disadvantage of SGD is that the gradients in different directions are scaled uniformly, resulting in limited convergence speed and sensitive choice of the learning rate, and thus has spawned a lot of recent interest in accelerating SGD from the algorithmic and practical perspectives.

Recently, many *adaptive* first-order optimization methods have been proposed to achieve rapid training progress with element-wise scaled learning rates, and we can only mention a few here due to space limits. In their pioneering work, Duchi et al. (2011) proposed AdaGrad, which scales the gradient by the square root of the accumulative square gradients from the first iteration. While AdaGrad works well for sparse settings, its performance significantly degrades for dense settings, primarily due to the monotonic increase of the accumulation. Subsequently, several methods have been proposed with the intuition to limit the accumulation to a small window of past iterations, and in particular exponentially reduce the weight of earlier iterations. Notable works incorporating this method are RMSProp (Tieleman & Hinton, 2012), AdaDelta (Zeiler, 2012), and Adam (Kingma & Ba, 2015), among which Adam has become the default optimization algorithm across many deep learning applications because of its fast convergence speed and relatively consistent selections of hyper-parameters (Ruder, 2016). However, it has been observed that these adaptive optimization methods may converge to bad/suspicious local optima, resulting in worse generalization ability than their non-adaptive counterparts (Wilson et al., 2017), or fail to converge due to unstable and extreme learning rates (Luo et al., 2019).

Quasi-Newton methods have been widely used in solving convex optimization problems, due to their efficient computation and fast convergence rate (Broyden, 1967; Dennis & Moré, 1977). However, the stochastic, high dimensional and nonconvex nature of many machine learning tasks, such as

training deep neural networks, has rendered many classical quasi-Newton methods ineffective and/or inefficient (Keskar & Berahas, 2016; Wang et al., 2017; Yao et al., 2020). Indeed, in many natural language processing (NLP) and computer vision (CV) tasks (He et al., 2016; Ma & Hovy, 2016; Luo et al., 2019), SGD (with momentum) is chosen as the optimizer, benefiting from its stable and efficient training and outstanding generalization.

In this work, we develop APOLLO, a quasi-Newton method for nonconvex stochastic optimization to simultaneously tackle the aforementioned challenges of stochastic variance, nonconvexity and inefficiency. Algorithmically, APOLLO dynamically incorporates the curvature of the objective function with diagonally approximated Hessian. It only requires first-order gradients and updates the approximation of the Hessian diagonally so that it satisfies a parameter-wise version of the weak secant condition (Wolfe, 1959). To handle nonconvexity, we replace the Hessian with its rectified absolute value, the computation of which is also efficient under our diagonal approximation, yielding an efficient optimization algorithm with linear complexity for both time and memory (§3). Experimentally, through three tasks on CV and NLP with popular deep neural networks, including ResNets (He et al., 2016), LSTMs (Hochreiter & Schmidhuber, 1997) and Transformers (Vaswani et al., 2017), we demonstrate that APOLLO significantly outperforms SGD and variants of Adam, in terms of both convergence speed and generalization performance (§4).

2 BACKGROUNDS

In this section we set up the notations on nonconvex stochastic optimization, briefly review the (quasi-) Newton methods, and discuss the problems of applying quasi-Newton methods to nonconvex stochastic optimization that we attempt to study in the rest of the paper.

2.1 NONCONVEX STOCHASTIC OPTIMIZATION

In this paper, we consider the following stochastic optimization problem:

$$\min_{\theta \in \mathcal{R}^d} f(\theta) = \mathbb{E}[l(\theta; \Gamma)] \tag{1}$$

where $l : \mathcal{R}^d \times \mathcal{R}^n \to \mathcal{R}$ is a continuously differentiable (and possible nonconvex) function, $\theta \in \mathbb{R}^d$ denotes the parameter to be optimized, $\Gamma \in \mathcal{R}^n$ denotes a random variable with distribution function P, and $\mathbb{E}[\cdot]$ denotes the expectation w.r.t Γ. Intuitively, Γ incorporates noises in f, leading to a stochastic objective function. A special case of (1) that arises frequently in machine learning is the empirical risk minimization problem:

$$\min_{\theta \in \mathcal{R}^d} f(\theta) = \frac{1}{N} \sum_{i=1}^{N} l_i(\theta) \tag{2}$$

where $l_i : \mathcal{R}^d \to \mathcal{R}$ is the loss function corresponds to the i-th data, and N is the number of data samples that is assumed to be extremely large. Objective functions may also have other sources of noise than data subsampling, such as dropout (Srivastava et al., 2014) in deep neural networks.

Decoupled Parameters. In this work, we consider a setting of decoupled parameters: $\theta = \{\theta^{(l)}, l = 1, \ldots, L\}$. Intuitively, under this setting the parameter θ is decoupled into a sequence of parameters serving different functionalities. For example, in neural network training the parameters of a neural network can be naturally decoupled into the parameters of different layers or modules.

2.2 NEWTON AND QUASI-NEWTON METHODS

Newton's method usually employs the following updates to solve (1):

$$\theta_{t+1} = \theta_t - H_t^{-1} g_t \tag{3}$$

where $g_t = \nabla f(\theta_t)$ is the gradient at θ_t and $H_t = \nabla^2 f(\theta_t)$ is the Hessian matrix. The convergence rate of Newton's method is *quadratic* under standard assumptions (Nocedal & Wright, 2006). However, one major challenge with this method is the expensive computation of the inverse Hessian at every iteration and the corresponding quadratic memory complexity.

A standard alternative to Newton's method is a class of quasi-Newton methods, which have been widely used in solving convex deterministic optimization:

$$\theta_{t+1} = \theta_t - \eta_t B_t^{-1} g_t \tag{4}$$

where η_t is the stepsize (a.k.a learning rate), B_t is an approximation to the Hessian matrix $\nabla^2 f(\theta_t)$ at θ_t, which is updated based on the well-known secant equation:

$$\begin{aligned} B_{t+1} &= \operatorname*{argmin}_B \|B - B_t\| \\ \text{s.t.} \quad & B_{t+1} s_t = y_t \quad \text{(secant equation)} \end{aligned} \tag{5}$$

where $s_t = \theta_{t+1} - \theta_t$ and $y_t = g_{t+1} - g_t$. B_{t+1} is, in the sense of some matrix norm, the closest to B_t among all symmetric matrices that satisfy the secant equation. Each choice of the matrix norm results in a different update formula, such as DFP (Davidon, 1991; Fletcher, 1987) and BFGS (Broyden, 1970; Fletcher, 1970; Goldfarb, 1970; Shanno, 1970). The popularity of this method is due to the fact that only the gradient of the objective function is required at each iteration. Since no second derivatives (Hessian) are required, quasi-Newton method is sometimes more efficient than Newton's method, especially when the computation of Hessian is expensive. To further reduce memory cost, one seminal work is the limited memory BFGS (L-BFGS) (Liu & Nocedal, 1989; Byrd et al., 1995) that achieves desirable linear computational and memory complexity by approximating the Hessian as a series of sum of first order information from previous iterations.

2.3 Problems of quasi-Newton Methods

Despite their impressive successes on convex deterministic optimization, quasi-Newton methods suffer from their own problems in the scenario of large-scale nonconvex stochastic optimization.

Stochastic Variance. One main challenge that prevents quasi-Newton methods from being applied to nonconvex stochastic optimization (1) is the variance introduced by the stochastic nature of the problem. At each iteration, only the stochastic gradient g_t is available, which is an unbiased estimation of the gradient $\nabla f(\theta_t)$ and may lead to an erroneous approximation of Hessian (Byrd et al., 2011).

Nonconvexity. Another key challenge in designing stochastic quasi-Newton methods for nonconvex problem lies in the difficulty of preserving the positive-definiteness of B_t in (5), due to the nonconvexity of the objective function. What is worse is that performing line search is infeasible in the stochastic setting, due to the presence of noise in the stochastic gradients (Wang et al., 2017).

Computational and Memory Efficiency. Even though quasi-Newton methods are more efficient than Newton's method, the time and memory complexities are still relatively large compared with adaptive first-order methods. For instance, L-BFGS requires to store first-order information from m previous iterations with commonly $m \geq 5$, which is still too expensive for deep neural networks containing millions of parameters. Moreover, adapting quasi-Newton methods to nonconvex stochastic optimization probably introduces additional computation, further slowing down these methods.

Due to the aforementioned problems, no quasi-Newton methods (to our best knowledge) designed for nonconvex optimization consistently outperform adaptive first-order algorithms w.r.t convergence speed and generalization performance. The main goal of this work is to algorithmically design and experimentally demonstrate a novel quasi-Newton method, in hope of improving the convergence speed and generalization performance of nonconvex stochastic optimization eventually.

3 Adaptive Parameter-Wise Diagonal Quasi-Newton

With the end goal of designing an efficient quasi-Newton method to solve the problem in (1) in mind, we first propose to approximate the Hessian with a diagonal matrix, whose elements are determined by the variational approach subject to the *parameter-wise* weak secant equation (§3.1). Then, we explain our stepsize bias correction technique to reduce the stochastic variance in §3.2. To handle nonconvexity, we directly use the rectified absolute value of the diagonally approximated Hessian as the preconditioning of the gradient (§3.3). The pseudo-code is shown in Algorithm 1.

3.1 Quasi-Newton Methods with Diagonal Hessian Approximation

As discussed in Bordes et al. (2009), designing an efficient stochastic quasi-Newton algorithm involves a careful trade-off between the sparsity of the approximation matrix B_t and the quality of its

approximation of the Hessian H_t, and diagonal approximation is a reasonable choice (Becker et al., 1988). If B is chosen to be a diagonal matrix to satisfy the secant equation in (5), one can obtain an updating formula similar to the SGD-QN algorithm (Bordes et al., 2009).

An alternative of the secant equation in the updating formula (5), as first introduced by Nazareth (1995), is the weak secant equation (Dennis & Wolkowicz, 1993):

$$\begin{aligned} B_{t+1} &= \operatorname*{argmin}_{B} \|B - B_t\| \\ \text{s.t.} \quad & s_t^T B_{t+1} s_t = s_t^T y_t \quad \text{(weak secant equation)} \end{aligned} \tag{6}$$

The motivation of using the weak secant condition in diagonal quasi-Newton method is straightforward: the standard mean-value theorem might not necessarily hold for vector-valued functions expressed in the secant equation, $B_{t+1}s_t = y_t \approx \nabla^2 f(\theta_t)s_t$. Thus, we do not know whether there exists a vector $\tilde{\theta} \in \mathbb{R}^d$ such that $y_t = \nabla^2 f(\tilde{\theta})s_t$ (Dennis & Moré, 1977). On the other hand, the Taylor theorem ensures that there exists such $\tilde{\theta}$ that $s_t^T y_t = s_t^T \nabla^2 f(\tilde{\theta}) s_t$, leading to the reasonable assumption of the weak secant condition (6).

Based on the variational technique in Zhu et al. (1999), the solution of (6) with Frobenius matrix norm is:

$$\Lambda \triangleq B_{t+1} - B_t = \frac{s_t^T y_t - s_t^T B_t s_t}{\|s_t\|_4^4} \operatorname{Diag}(s_t^2) \tag{7}$$

where s_t^2 is the element-wise square vector of s_t, $\operatorname{Diag}(s_t^2)$ is the diagonal matrix with diagonal elements from vector s_t^2, and $\|\cdot\|_4$ is the 4-norm of a vector.

Parameter-Wise Weak Secant Condition. However, in optimization problems with high-dimensional parameter space, such as training deep neural networks with millions of parameters, the weak secant condition might be too flexible to produce a good Hessian approximation. In the setting of decoupled parameters (§2.1), we propose a parameter-wise version of the weak secant equation to achieve a trade-off between the secant and weak secant conditions: for each parameter $\theta^{(l)} \in \theta$, we update B corresponding to $\theta^{(l)}$ by solving (6) individually. Remarkably, the secant condition restricts B with an equation of a d-dimensional vector, while the weak secant condition relaxes it with a 1-dimensional scalar. The parameter-wise weak secant condition expresses the restriction as a l-dimension vector ($1 < l < d$), resulting in a reasonable trade-off. The updating formula is the same as (7) for each parameter-wise B.

3.2 Stepsize Bias Correction

To mitigate the stochastic variance problem in stochastic quasi-Newton methods, APOLLO utilizes stepsize bias correction on the stochastic gradients at each step t. Concretely, comparing the parameter updating formulas of Newton's method (3) and quasi-Newton methods (4), we observe that the optimal stepsize η_t equals to 1 if the Hessian approximation B_t and the stochastic gradient g_t are close to the exact Hessian H_t and gradient $\nabla f(\theta_t)$, respectively. Inspired by this, we correct the stepsize bias in the stochastic gradient g_t by replacing it with a corrected gradient $g'_t = \eta_t g_t$. Together with the corresponding corrected $y'_t = g'_{t+1} - g'_t = \eta_t y_t$, we correct the updating term Λ of B_t in (7) by:

$$\begin{aligned} \Lambda' &= \frac{s_t^T y'_t - s_t^T B_t s_t}{\|s_t\|_4^4} \operatorname{Diag}(s_t^2) \\ &= -\frac{d_t^T y_t + d_t^T B_t d_t}{\|d_t\|_4^4} \operatorname{Diag}(d_t^2) \end{aligned} \tag{8}$$

where $d_t = -s_t/\eta_t = B_t^{-1} g_t$ is the update direction. Based on previous studies, incorporating moving averages for the stochastic gradients significantly reduces the variance (Johnson & Zhang, 2013; Kingma & Ba, 2015). We follow these work and apply exponential moving average to g_t, together with the initialization bias correction:

$$m_{t+1} = \frac{\beta(1-\beta^t)}{1-\beta^{t+1}} m_t + \frac{1-\beta}{1-\beta^{t+1}} g_{t+1} \tag{9}$$

where $0 < \beta < 1$ is the decay rate of the exponential moving average. Note that we do not apply moving average methods to the approximated Hessian, though the diagonal matrix is easier to be explicitly formed to average than full matrices. Further improving APOLLO by investigating the moving average of the diagonal matrix B_t might be a interesting direction of future work.

Algorithm 1: APOLLO, our proposed algorithm for nonconvex stochastic optimization. All operations on vector are element-wise. Good default settings are $\beta = 0.9$ and $\epsilon = 1e^{-4}$.

Initial: $m_0, d_0, B_0 \leftarrow 0, 0, 0$ `// Initialize` m_0, d_0, B_0 `to zero`
while $t \in \{0, \ldots, T\}$ **do**
  for $\theta \in \{\theta^1, \ldots, \theta^L\}$ **do**
    $g_{t+1} \leftarrow \nabla f_t(\theta_t)$ `// Calculate gradient at step` t
    $m_{t+1} \leftarrow \frac{\beta(1-\beta^t)}{1-\beta^{t+1}} m_t + \frac{1-\beta}{1-\beta^{t+1}} g_{t+1}$ `// Update bias-corrected moving`
    $\alpha \leftarrow \frac{d_t^T(m_{t+1}-m_t)+d_t^T B_t d_t}{(\|d_t\|_4+\epsilon)^4}$ `// Calculate coefficient of` B `update`
    $B_{t+1} \leftarrow B_t - \alpha \cdot \mathrm{Diag}(d_t^2)$ `// Update diagonal Hessian`
    $D_{t+1} \leftarrow \mathrm{rectify}(B_{t+1}, 1)$ `// Handle nonconvexity`
    $d_{t+1} \leftarrow D_{t+1}^{-1} m_{t+1}$ `// Calculate update direction`
    $\theta_{t+1} \leftarrow \theta_t - \eta_{t+1} d_{t+1}$ `// Update parameters`
  end
end
return θ_T

3.3 RECTIFIED ABSOLUTE VALUE OF HESSIAN FOR NONCONVEXITY

To guarantee convergence, quasi-Newton methods require the approximated Hessian matrix B_t to be positive definite at each step. The common strategy in previous studies is to solve the updating formula in (5) by restricting the candidate matrix B to be symmetric positive definite. It is known that the BFGS update preserves the positive-definiteness of B_{t+1} as long as the curvature condition $s_t^T y_t > 0$ holds, which can be guaranteed for strongly convex problem. For nonconvex problem, the curvature condition can be satisfied by performing a line search, which is, however, expensive or even infeasible in stochastic setting, because the exact function values and gradient information are unavailable. Wang et al. (2017) proposed the stochastic damped L-BFGS (SdLBFGS) method that implicitly generates a positive definite matrix without line search.

To handle nonconvexity, we adopt a different strategy that does not require the solution of B_t in (5) to be positive definite. Intuitively, we search for B_t that is a good approximation of the real Hessian, which is unnecessarily positive definite in nonconvex problem. When we use B_t as preconditioning to calculate the update direction, we use its absolute value:

$$|B_t| = \sqrt{B_t^T B_t} \tag{10}$$

where $\sqrt{\cdot}$ is the positive definite square root of a matrix. The motivation of absolute value is straightforward: for dimensions with large absolute values of curvature, the objective function could be very sharp and we would prefer to take relatively smaller steps than those flatter dimensions. Since APOLLO formulate B_t as a diagonal matrix, the cost of computing $|B_t|$ is marginal.

Rectified Absolute Value of B_t For nonconvex objective functions, there exist inflection points whose curvatures are zero. To prevent the steps from becoming arbitrarily large, we rectify the absolute value of B_t with a convexity hyper-parameter σ:

$$D_t = \mathrm{rectify}(B_t, \sigma) = \max(|B_t|, \sigma) \tag{11}$$

where the $\mathrm{rectify}(\cdot, \sigma)$ function is similar to the rectified linear unit (ReLU) (Nair & Hinton, 2010) with threshold set to σ. The update direction in (8) is then $d_t = D_t^{-1} m_t$.

AdaHessian (Yao et al., 2020) used an idea similar to the absolute values of B_t to handle nonconvexity, where the root mean square averaging is applied to compute the Hessian diagonal. Different from APOLLO, AdaHessian requires second-order information to compute the Hessian matvec oracle and approximate the Hessian diagonal using Hutchinson's method, which is significantly more costly.

3.4 INITIALIZATION

APOLLO initializes the moving average of gradient m_0, the parameter update direction d_0 and the diagonal approximation of Hessian B_0 as (vector of) zeros. In the following, we show that this zero

initialization approach leads to coupled stepsize η and convexity σ, allowing us to eliminate one hyper-parameter of η or σ.

Coupled Stepsize η and Convexity σ. With the zero initialization of $m_0,\ d_0,$ and B_0, we have the following theorem to illustrate the relation between η and σ (details in Appendix A):

Theorem 1. *Given zero initialization of $m_0,\ d_0,$ and B_0 and a fixed parameter intialization θ_0. Suppose that we have two sets of hyper-parameters η, σ and η', σ' with the same ratio: $\frac{\eta}{\sigma} = \frac{\eta'}{\sigma'}$. Then the convergence trajectories of these two sets of hyper-parameters are exactly the same:*

$$\theta_t = \theta'_t,\ \forall t \in \{1, \ldots, T\}. \tag{12}$$

where θ_t and θ'_t are the parameters of (η, σ) and (η', σ') at iteration t, respectively.

From Theorem 1, we observe that η and σ are coupled with each other and in practice we only need to tune one of them, leaving the other fixed. Therefore, in our experiments (§4), we fix $\sigma = 1$ and tune η on different problems.

Learning Rate Warmup for APOLLO As discussed in Kingma & Ba (2015), zero initialization leads to estimations biased towards zero in the initial iterations. For the moving average m_t, this bias can be corrected by dividing the bias-correction term (9). For d_t and B_t, however, we cannot derive such bias correction terms. Fortunately, a simple warmup heuristic of η at the beginning iterations achieves remarkablely stable training in our experiments.

4 EXPERIMENTS

To evaluate APOLLO, we conduct experiments on four benchmark datasets across three tasks of CV and NLP that are commonly used to evaluate optimization algorithms: CIFAR-10 (Krizhevsky & Hinton, 2009) and ImageNet (Deng et al., 2009) for image classification; One Billion Words (Chelba et al., 2013) for language modeling; and WMT 2014 English-German for neural machine translation. The three baseline methods we compare with are SGD with momentum (Bottou & Bousquet, 2008), Adam (Kingma & Ba, 2015), and Rectified Adam (RAdam) (Liu et al., 2020). Following Loshchilov & Hutter (2019), we decouple weight decays in Adam and RAdam in all the experiments. For each experiment, we report the average over 5 runs. More detailed descriptions, results and analysis of the conducted experiments are provided in Appendix B.

4.1 IMAGE CLASSIFICATION

We begin our experiments with an evaluation of the convergence and generalization performance on image classification. The results on CIFAR-10 and ImageNet are presented in Figure 1 and Table 1, together with the three baselines. For each optimizer, we use two scheduling strategies of learning rate decay: i) milestone that decays the learning rate at the end of some predefined epochs; and cosine annealing schedule proposed in Loshchilov & Hutter (2017) (details in Appendix B.1).

Table 1: Test accuracy on CIFAR-10 and ImageNet.

Method	CIFAR-10 milestone	CIFAR-10 cosine	ImageNet milestone	ImageNet cosine
SGD	93.91	94.53	77.19	78.17
Adam	91.41	91.56	71.72	71.19
RAdam	91.80	91.88	72.37	71.64
Adam-adj	93.74	94.24	76.86	77.54
RAdam-adj	93.88	94.38	76.91	77.68
APOLLO	**94.20**	**94.60**	**77.90**	**78.54**

Though Loshchilov & Hutter (2019) claimed that the optimal settings of the learning rate and weight decay factor in Adam with decoupled weight decay is more independent than the original Adam, we observed that the strength of weight decay regularization is still co-related with the learning rate. Since the strength of regularization has significant effect on both the performance of convergence and generalization, we adjust the weight decay rates for Adam and RAdam, named Adam-adj and RAdam-adj, so that the they have similar strength of regularization for fair comparison. More detailed analysis on the effect of weight decay is provided in Appendix C.2.

From Figure 1 and Table 1, we first see that Adam and RAdam, without adjusting the weight decay rates, converge much faster than other optimization methods, while obtaining significantly worse classification accuracy. After adjusting the weight decay rates, the test accuracy of Adam-adj and RAdam-adj remarkably improves, with rapid decline of convergence speed. This suggests that the

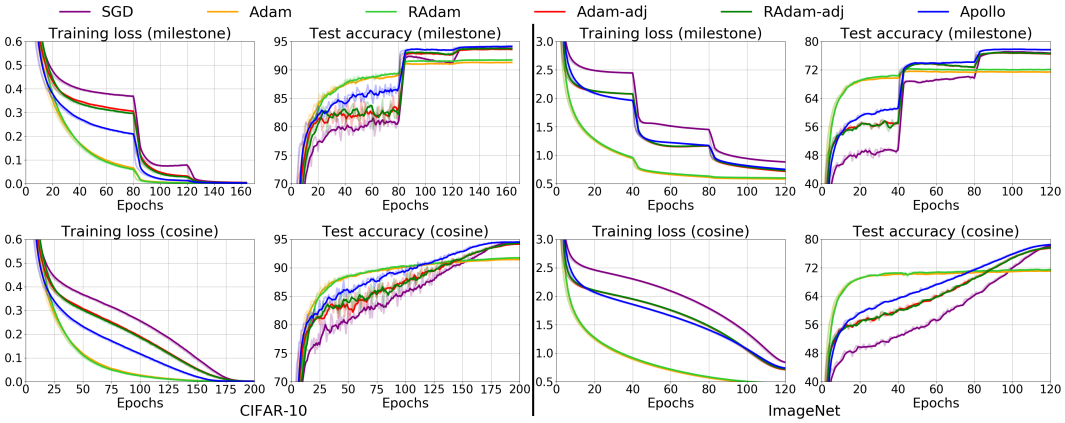

Figure 1: Training loss and test accuracy of ResNet-110 on CIFAR-10 and ResNeXt-50 on ImageNet, with two schedule strategies of learning rate decay.

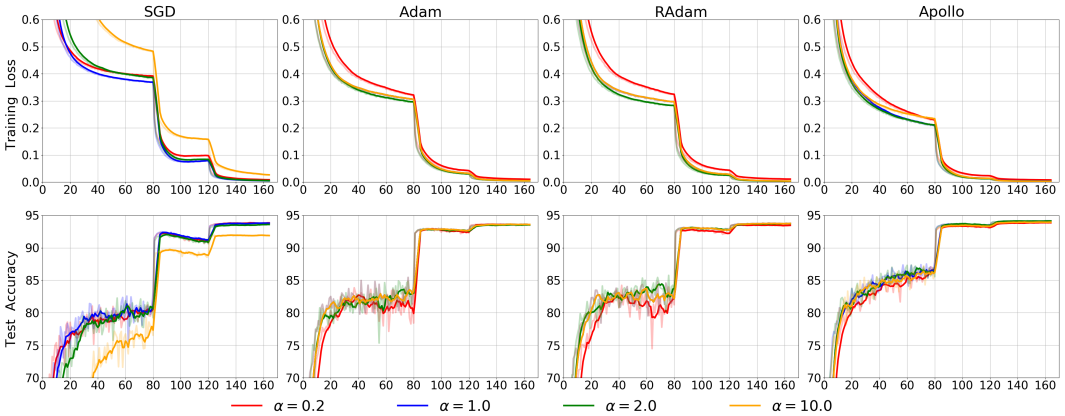

Figure 2: SGD, Adam, RAdam and APOLLO with different learning rates on CIFAR-10.

fast convergence speed of Adam and RAdam results from relatively weak regularization. Thus, the effect of regularization strength needs to be considered when we analyze the performance of different optimization methods. Second, APOLLO outperforms all the three baselines on both the convergence speed and classification accuracy, demonstrating its effectiveness on training the ResNet architectures based on convolutional neural networks (CNNs) (LeCun et al., 1989).

Robustness to Learning Rate Change. Besides performance improvements, we also investigate the robustness of different optimization methods to the change of learning rate. For each optimizer, we use the learning rate in the previous experiment (Table 1) as the base, i.e. 0.1 for SGD, 0.001 for Adam and RAdam, and 0.5 for APOLLO. Then, we explore different learning rates that are α times of the base learning rate, with $\alpha \in \{0.2, 1.0, 2.0, 10.0\}$. To eliminate the impact of weight decay, we adjust the weight decay rates according to the factor α. Experimental results with ResNet-110 on CIFAR-10 are summarized in Figure 2. After correcting the impact of weight decay, all the optimization methods, except SGD with $\alpha = 10.0$, achieves consistent model performance (highly overlapped training and test curves with each other), while APOLLO slightly improves the robustness of model training over the three baseline methods.

4.2 LANGUAGE MODELING

To evaluate APOLLO on Recurrent Neural Networks (RNNs) that are applied in a wide range of NLP tasks (Graves, 2013), we conduct experiments on the One Billion Words dataset, using a two-layer LSTM network for language modeling (details in Appendix B.2).

Figure 3 and Table 2 illustrate the perplexity (PPL) of training and test for each optimization method. As shown in Figure 3, although APOLLO is slower than Adam and RAdam in the first few updates, its convergence is much faster after that. On generalization performance, APOLLO achieves significant

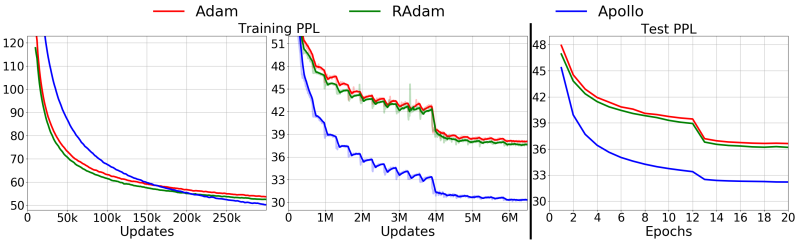

Figure 3: Language modeling (LSTMs) on One Billion Words.

Method	PPL
Adam	36.68
RAdam	36.20
APOLLO	**32.21**

Table 2: Test PPL.

improvements (around 4.0 PPL points on test data) over Adam and RAdam. This demonstrates the effectiveness of APOLLO on training LSTM-based neural architectures.

4.3 NEURAL MACHINE TRANSLATION

To evaluate APOLLO on Attention-based Transformer architecture (Vaswani et al., 2017), we train the Transformer-base model on the WMT2014 English-German (EN-DE) dataset (around 4.5M sentence pairs), using Adam, RAdam and APOLLO. We use the same data preprocessing steps as in Ma et al. (2019) (details in Appendix B.3). Table 3 provides the BLEU scores of APOLLO, together with Adam and RAdam as baselines. For each experiment, we report the mean and standard variance over 5 runs. APOLLO obtains improvements over Adam and RAdam on BLEU scores (0.49 better than Adam and 0.21 than RAdam).

Table 3: BLEU scores on WMT-14.

Method	BLEU
Adam	27.83±0.06
RAdam	28.11±0.17
APOLLO	**28.32**±0.12

5 RELATED WORK

Stochastic Quasi-Newton Methods. In the literature of (nonconvex) stochastic quasi-Newton methods, several algorithms have been developed recently for large-scale machine learning problems: oLBFGS (Schraudolph et al., 2007; Mokhtari & Ribeiro, 2015), RES (Mokhtari & Ribeiro, 2014), SFO (Sohl-Dickstein et al., 2014), SQN (Byrd et al., 2016), SdLBFGS (Wang et al., 2017), and AdaQN (Keskar & Berahas, 2016), among which only SdLBFGS and AdaQN are designed to solve nonconvex optimization problems. The SdLBFGS algorithm carefully controls the quality of modified BFGS updates to preserve the positive-definiteness of B_t in (5) without line search, while AdaQN shares a similar idea but specifically designed for RNNs.

Adaptive First-Order Methods. From the diagonal approximation of Hessian, APOLLO is also related to those diagonally-scaled first-order algorithms, such as AdaGrad (Duchi et al., 2011), RMSProp (Tieleman & Hinton, 2012), AdaDelta (Zeiler, 2012), and Adam (Kingma & Ba, 2015). Subsequently, a number of techniques have emerged to theoretically justify and algorithmically improve Adam, including AMSGrad (Reddi et al., 2018), AdaBound (Luo et al., 2019) and RAdam (Liu et al., 2020). The main difference is that the diagonal preconditioning in APOLLO is directly derived from the quasi-Newton updating formula (6), which is closely related to the Hessian of the objective.

Stochastic Second-Order Hessian-Free Methods. Stochastic Second-Order Hessian-Free methods (Martens, 2010; Martens & Sutskever, 2011) implicitly solve quadratic models using matrix-vector products. K-FAC (Martens & Grosse, 2015) computes a second-order step by constructing an invertible approximation of the Fisher information matrix in an online fashion. Shampoo (Gupta et al., 2018) approximates the Fisher information matrix using low-rank decomposition. Recently, Yao et al. (2020) proposed AdaHessian, which approximates the Hessian diagonal using Hutchinson's method. These second-order methods differ from APOLLO mainly in the request of second-order information of the objective function at each iteration.

6 CONCLUSION

We have introduced APOLLO, a simple and computationally efficient quasi-Newton algorithm for nonconvex stochastic optimization. This method is aimed towards large-scale optimization problems in the sense of large datasets and/or high-dimensional parameter spaces, such as machine learning with deep neural networks. Experimental results on three CV and NLP tasks demonstrate the effectiveness of APOLLO, in terms of both convergence speed and generalization performance.

REFERENCES

Sue Becker, Yann Le Cun, et al. Improving the convergence of back-propagation learning with second order methods. In *Proceedings of the 1988 connectionist models summer school*, pp. 29–37, 1988.

Antoine Bordes, Léon Bottou, and Patrick Gallinari. SGD-QN: Careful quasi-newton stochastic gradient descent. *The Journal of Machine Learning Research*, 10:1737–1754, 2009.

Léon Bottou and Olivier Bousquet. The tradeoffs of large scale learning. In *Advances in neural information processing systems*, pp. 161–168, 2008.

Charles G Broyden. Quasi-newton methods and their application to function minimisation. *Mathematics of Computation*, 21(99):368–381, 1967.

Charles George Broyden. The convergence of a class of double-rank minimization algorithms. *IMA Journal of Applied Mathematics*, 6(1):76–90, 1970.

Richard H Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on scientific computing*, 16(5):1190–1208, 1995.

Richard H Byrd, Gillian M Chin, Will Neveitt, and Jorge Nocedal. On the use of stochastic hessian information in optimization methods for machine learning. *SIAM Journal on Optimization*, 21(3): 977–995, 2011.

Richard H Byrd, Samantha L Hansen, Jorge Nocedal, and Yoram Singer. A stochastic quasi-newton method for large-scale optimization. *SIAM Journal on Optimization*, 26(2):1008–1031, 2016.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*, 2013.

William C Davidon. Variable metric method for minimization. *SIAM Journal on Optimization*, 1(1): 1–17, 1991.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

John E Dennis, Jr and Jorge J Moré. Quasi-newton methods, motivation and theory. *SIAM review*, 19 (1):46–89, 1977.

John E Dennis, Jr and Henry Wolkowicz. Sizing and least-change secant methods. *SIAM Journal on Numerical Analysis*, 30(5):1291–1314, 1993.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

Roger Fletcher. A new approach to variable metric algorithms. *The computer journal*, 13(3), 1970.

Roger Fletcher. *Practical methods of optimization*. John Wiley & Sons, 1987.

D Goldfarb. A family of variable metric updates derived by variational means. *Mathematics of Computation*, 24(109):23–26, 1970.

Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint:1308.0850*, 2013.

Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. *arXiv preprint arXiv:1802.09568*, 2018.

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

Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97, 2012.

Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

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

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in neural information processing systems*, pp. 315–323, 2013.

Nitish Shirish Keskar and Albert S Berahas. AdaQN: An adaptive quasi-newton algorithm for training rnns. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 1–16. Springer, 2016.

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

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989.

Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. In *International Conference on Learning Representations*, 2020.

Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Liangchen Luo, Yuanhao Xiong, Yan Liu, and Xu Sun. Adaptive gradient methods with dynamic bound of learning rate. In *International Conference on Learning Representations*, 2019.

Xuezhe Ma and Eduard Hovy. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1064–1074, Berlin, Germany, August 2016.

Xuezhe Ma, Chunting Zhou, Xian Li, Graham Neubig, and Eduard Hovy. Flowseq: Non-autoregressive conditional sequence generation with generative flow. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, pp. 4282–4292, Hong Kong, November 2019.

James Martens. Deep learning via hessian-free optimization. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, pp. 735–742, 2010.

James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pp. 2408–2417, 2015.

James Martens and Ilya Sutskever. Learning recurrent neural networks with hessian-free optimization. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 1033–1040. Citeseer, 2011.

Aryan Mokhtari and Alejandro Ribeiro. Res: Regularized stochastic bfgs algorithm. *IEEE Transactions on Signal Processing*, 62(23):6089–6104, 2014.

Aryan Mokhtari and Alejandro Ribeiro. Global convergence of online limited memory bfgs. *The Journal of Machine Learning Research*, 16(1):3151–3181, 2015.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.

JL Nazareth. If quasi-newton then why not quasi-cauchy. *SIAG/Opt Views-and-news*, 6:11–14, 1995.

Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. FairSeq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pp. 48–53, 2019.

Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1): 145–151, 1999.

Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. In *International Conference on Learning Representations*, 2018.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pp. 400–407, 1951.

Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

Nicol N Schraudolph, Jin Yu, and Simon Günter. A stochastic quasi-newton method for online convex optimization. In *Artificial intelligence and statistics*, pp. 436–443, 2007.

David F Shanno. Conditioning of quasi-newton methods for function minimization. *Mathematics of computation*, 24(111):647–656, 1970.

Jascha Sohl-Dickstein, Ben Poole, and Surya Ganguli. Fast large-scale optimization by unifying stochastic gradient and quasi-newton methods. In *International Conference on Machine Learning*, pp. 604–612, 2014.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

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*, 4(2), 2012.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Xiao Wang, Shiqian Ma, Donald Goldfarb, and Wei Liu. Stochastic quasi-newton methods for nonconvex stochastic optimization. *SIAM Journal on Optimization*, 27(2):927–956, 2017.

Ashia C Wilson, Rebecca Roelofs, Mitchell Stern, Nati Srebro, and Benjamin Recht. The marginal value of adaptive gradient methods in machine learning. In *Advances in neural information processing systems*, pp. 4148–4158, 2017.

Philip Wolfe. The secant method for simultaneous nonlinear equations. *Communications of the ACM*, 2(12):12–13, 1959.

Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1492–1500, 2017.

Zhewei Yao, Amir Gholami, Sheng Shen, Kurt Keutzer, and Michael W Mahoney. ADAHESSIAN: An adaptive second order optimizer for machine learning. *arXiv preprint arXiv:2006.00719*, 2020.

Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint:1212.5701*, 2012.

Mingfa Zhu, John Lawrence Nazareth, and Henry Wolkowicz. The quasi-cauchy relation and diagonal updating. *SIAM Journal on Optimization*, 9(4):1192–1204, 1999.

APPENDIX: APOLLO: AN ADAPTIVE PARAMETER-WISE DIAGONAL QUASI-NEWTON METHOD FOR NONCONVEX STOCHASTIC OPTIMIZATION

A COUPLED STEPSIZE AND CONVEXITY

Before proving Theorem 1, we first define the notations.

Let $\alpha = \frac{\eta'}{\eta} = \frac{\sigma'}{\sigma}$ be the ratio of the two sets of learning rates. Let $m'_t,\ d'_t$ and B'_t be the corresponding terms of parameter θ'_t at step t for (η', σ').

Proof of Theorem 1

Proof. Induction on the step of updates t, we attempt to prove that at each step t:

$$m_t = m'_t, \quad B'_t = \alpha B_t, \text{ and } \theta_t = \theta'_t, \ \forall t \tag{13}$$

Initial step: when $t = 1$, since $\theta_0 = \theta'_0$, we have $m_1 = m'_1$. With $d_0 = d'_0 = 0$ and (8), we have $B_1 = B'_1 = 0$ and:

$$\begin{aligned} D_1 &= \text{recify}(B_1, \sigma) = \sigma \\ D'_1 &= \text{recify}(B'_1, \sigma') = \sigma' \end{aligned}$$

Then, $D'_1 = \alpha D_1$ and

$$\theta'_1 = \theta'_0 - \eta' D'^{-1}_1 m'_1 = \theta_0 - \eta\alpha(D_1^{-1}/\alpha)m_1 = \theta_0 - \eta D_1^{-1} m_1 = \theta_0.$$

Thus, the statement (13) is true.

Induction on step t: Suppose that the statement (13) is true for all the previous t steps. Now we prove the case $t + 1$. From the inductive assumption and (9), we have,

$$B'_t = \alpha B_t, \quad d'_t = \frac{1}{\alpha} d_t \text{ and } m_{t+1} = m'_{t+1}.$$

From (8),

$$\begin{aligned} B'_{t+1} &= B'_t - \frac{{d'}_t^T {y'}_t + {d'}_t^T B'_t d'_t}{\|d'_t\|_4^4} \text{ Diag}({d'}_t^2) \\ &= \alpha B_t - \frac{(\frac{1}{\alpha} d_t)^T y_t + (\frac{1}{\alpha} d_t)^T (\alpha B_t)(\frac{1}{\alpha} d_t)}{\|(\frac{1}{\alpha} d_t)\|_4^4} \text{ Diag}((\frac{1}{\alpha} d_t)^2) \\ &= \alpha B_t - \alpha \frac{d_t^T y_t + d_t^T B_t d_t}{\|d_t\|_4^4} \text{ Diag}(d_t^2) \\ &= \alpha B_{t+1}. \end{aligned}$$

Then,

$$\begin{aligned} D'_{t+1} &= \text{recify}(B'_{t+1}, \sigma') \\ &= \text{recify}(\alpha B_{t+1}, \alpha\sigma) \\ &= \alpha \text{recify}(B_{t+1}, \sigma) \\ &= \alpha D_{t+1} \end{aligned}$$

and we have $\theta'_{t+1} = \theta_{t+1}$.

Finally, to sum up with the induction, we have proven Theorem 1. □

B EXPERIMENTAL DETAILS

B.1 IMAGE CLASSIFICATION

For CIFAR-10 dataset, we use the ResNet-100 architecture in the public implementation[1]. For ImageNet, we used the neural architecture of ResNeXt-50 (Xie et al., 2017). Batch size is 128 for CIFAR-10 and 256 for ImageNet. For each optimizer, we used two learning rate decay strategies. First, we train the model on CIFAR-10 for 164 epochs and decay the learning rate at the end of 80-th and 120-th epochs by 0.1. For ImageNet, we train the model for 120 epochs and decay the learning rate at at the end of 40-th and 80-th epochs by 0.1. Second, we also used the cosine annealing schedule (Loshchilov & Hutter, 2017). For the cosine annealing schedule, we train a CIFAR-10 model for 200 epochs and a ImageNet model for 120 epochs. The start learning rate is 0.1 for SGD, 0.001 for Adam and RAdam, and 0.5 for APOLLO. The weight decay rates for SGD and APOLLO are $5e^{-4}$ for CIFAR-10 and $2e^{-4}$ for ImageNet. For Adam and RAdam, we adjusted the weight decay rates based on the ratio of the learning rate, i.e. multiplying 500 to the original decay rates[2]. For APOLLO learning rates are warmed up linearly from 0.01 to 0.5 in the first 100 updates. Random cropping and random horizontal flipping are applied to training data. For each experiment, we conduct training on one NVIDIA Tesla V100 GPU for CIFAR-10 and eight NVIDIA Tesla V100 GPUs for ImageNet.

B.2 LANGUAGE MODELING

One Billion Words dataset (Chelba et al., 2013) is a publicly available benchmark for measuring progress of language modeling. It contains about 0.8 billion tokens with a vocabulary of 793,471 words, including sentence boundary markers. Different from Liu et al. (2020) which shrinks the vocabulary to about 0.64 million words, we used the standard vocabulary[3]. For the language model, we used two-layer LSTM with 2048 hidden states with adaptive softmax and 300-dimensional word embeddings as input. The cut-offs of the adaptive softmax are set to $[60000, 100000, 640000]$, which is different from Liu et al. (2020). Dropout (Srivastava et al., 2014) is applied to each layer with drop rate of 0.1. For Adam and RAdam, the learning rate is 0.001. For APOLLO, the learning rate is set to 10. Gradient clips with 1.0 are applied to all the optimization methods. Each model is trained for 20 epochs, and the learning rate decays at the end of the 12-th and 18-th epochs by decay rate 0.1. LSTMs are unrolled for 20 steps without resetting the LSTM states and the batch size is set to 128. Every models is trained on one NVIDIA Titan RTX GPU.

B.3 NEURAL MACHINE TRANSLATION

Our experiments on WMT 2014 English-German are based on the Transformer-base model (Vaswani et al., 2017), with implementation from the FairSeq package (Ott et al., 2019). The dataset is pre-processed following (Ma et al., 2019), using the scripts from FairSeq package. Specifically, we use word embedding with 512 dimension and 6-layer encoder/decoder with 8 multi-head attention and 2048 feed-forward dimensions. We apply 0.1 label smoothing (Szegedy et al., 2016), and perform totally $500,000$ updates to train each model. For Adam, we use start learning rate 0.0005 and the inverse square root decay (Vaswani et al., 2017). For RAdam and APOLLO, the start learning rates are 0.0005 and 10.0, respectively, and decays at the $250,000$ and $450,000$ updates by decay rate 0.1. As to the wamrup strategy, we use a linear warmup for Adam in the first 4000 updates, and the first 1000 updates for APOLLO. No warmup is applied to RAdam. For Adam we set $\beta = (0.9, 0.98)$, while for RAdam we set $\beta = (0.9, 0.999)$. Gradient clips with 1.0 are applied to all the optimization methods. The decoding beam size is set to 5, and the checkpoints of the last 10 epochs are averaged before evaluation. For each experiment, we conduct training on eight NVIDIA Tesla V100 GPUs with maximum batch size as 8192 tokens, apply dropout with a ratio 0.1, using weight decay of 0.0001[4].

[1] https://github.com/bearpaw/pytorch-classification

[2] Weight decay adjustment: $\gamma_{adam} * \eta_{adam} = \gamma_{apollo} * \eta_{apollo}$, where γ is the weight decay factor.

[3] https://github.com/rafaljozefowicz/lm/blob/master/1b_word_vocab.txt

[4] $1e^{-8}$ for APOLLO

Table 4: Classification accuracy on CIFAR-10 and ImageNet. For each experiment, we report the mean and standard variance over 5 runs.

Method	CIFAR-10 milestone decay	CIFAR-10 cosine annealing	ImageNet milestone decay	ImageNet cosine annealing
SGD	93.91±0.07	94.53±0.27	77.19±0.07	78.17±0.06
Adam	91.41±0.30	91.56±0.19	71.72±0.13	71.19±0.10
RAdam	91.80±0.04	91.88±0.15	72.37±0.08	71.64±0.14
Adam-adj	93.74±0.15	94.24±0.09	76.86±0.06	77.54±0.16
RAdam-adj	93.88±0.11	94.38±0.25	76.91±0.07	77.68±0.08
APOLLO	**94.20**±0.12	**94.60**±0.06	**77.90**±0.06	**78.54**±0.09

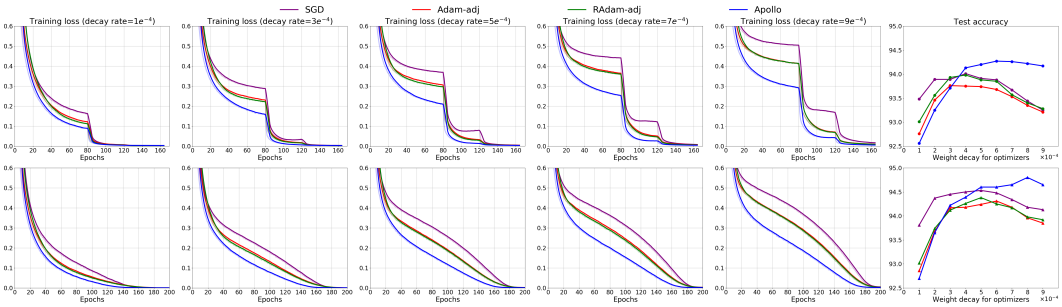

Figure 4: Training loss and test accuracy of ResNet-110 on CIFAR-10 with various rates of weight decay, with two schedule strategies of learning rate decay.

C DETAILED EXPERIMENTAL RESULTS

In this section, we report the detailed experimental results in Section 4, and the results the investigation of the effect of weight decay.

C.1 DETAILED RESULTS ON IMAGE CLASSIFICATION

Table 4 illustrates the details of the experimental results on Image Classification. For each experiment, we report the mean values with corresponding standard deviations over 5 runs.

C.2 EFFECT OF WEIGHT DECAY RATE ON OPTIMIZATION

To further investigate the effect of weight decay rate on converge speed and generalization performance for different optimization methods, we conduct experiments on CIFAR-10 of ResNet-110 with a range of weight decay rates from $1e^{-4}$ to $9e^{-4}$. As discussed before, for Adam and RAdam, we adjust the weight decay rate based on the ratio of the learning rate such that the they have similar strength of regularization for fair comparison:

$$\gamma_{adam} * \eta_{adam} = \gamma_{apollo} * \eta_{apollo} \tag{14}$$

where γ is the weight decay rate and η is the learning rate.

Figure 4 shows the convergence of different optimization methods with various rates of weight decay, together with the classification accuracy. APOLLO achieves improvements over all the three baselines on convergence speed with different rates of weight decay. For classification accuracy, APOLLO obtains the best accuracy when the weight decay rate is larger than $4e^{-4}$. When the weight decay rate is decreasing, SGD obtains the best accuracy, while APOLLO achieves comparable performance.