A Comparison of Efficient Optimization Methods for Fitting Neural Networks

Roman Kouznetsov, Pramit Das, Luke Francisco

April 2021

1 Introduction

This paper explores the mathematical details and performance of various popular optimization methods in deep learning, including SGD, AdaGrad, RMSProp, Adam, AdaMax, and AdaBelief. We first review the mathematical details of these algorithms and then implement them on the popular MNIST and CIFAR-10 data sets to evaluate their performance. Thereafter, we focus on the the convergence behavior of the methods. We also mention some relevant theorems in this context.

2 Algorithms for Optimization of Loss Functions

For this report, we shall consider image classification problems. We take f to be the widely-used cross-entropy loss and α to be a step size parameter. θ refers to the vector whose entries consist of the weights and biases of the neural network. Following is a discussion of common gradient descent algorithms used to estimate θ for neural networks in classification problems.

2.1 Stochastic Gradient Descent (SGD)

It is often difficult to compute the gradient on an entire data set due to either size or complexity. In SGD, we approximate the gradient by computing it based on a smaller subset (known as a minibatch) of the data. We update the parameter using the rule $\theta_{t+1} = \theta_t - \alpha \nabla f_b(\theta_t)$, where the gradient is approximated based on minibatch b. We then iterate over our minibatches until convergence. This approach often works better than vanilla gradient descent, as it uses information from different subsets of the data. Details have been discussed in [2].

2.2 AdaGrad

AdaGrad, originally proposed by Duchi et al. in 2010 [4], is the earliest modification to standard SGD we will consider. This algorithm iterates over minibatches (or the full data set) applying the update rule $\theta_{t+1} = \theta_t - \alpha \frac{g_t}{\sum_{j=1}^{t} g_j^2}$ until convergence, where g_t is the estimate of the gradient at the t^{th} step. AdaGrad's main improvement over SGD is its ability to use different learning rates depending on

the iteration. Here we use a time-dependent $\alpha_t = \alpha.t^{-1/2}$ to dampen the effect. Details have been discussed in [8].

2.3 RMSProp

RMSProp was first introduced by Geoffrey Hinton during a lecture [5]. It uses the update rule $\theta_{t+1} = \theta_t - \alpha\frac{g_t}{\sqrt{s_t}+\epsilon}$, where s_t is the exponential moving average (EMA) estimate of the squared gradient $s_t = \beta s_{t-1} + (1-\beta)g_t^2$. This helps prevent learning rates from rapidly converging to 0 as in AdaGrad, because the denominator of the update formula here is not a pure sum of squared gradients [8]. Additionally, this method is well-suited for use with minibatches in SGD because the EMA term prevents us from basing our update too much on the current gradient [5].

2.4 Adam

Adam [6] is a moment-based stochastic optimization algorithm which builds on AdaGrad and RMSProp. It proceeds with the following steps:

Initialize $t \leftarrow 0, m_0, v_0$
do
$t \leftarrow t + 1$
$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$
$m_t \leftarrow \beta_1 m_{t-1} + (1-\beta_1)g_t$
$v_t \leftarrow \beta_2 v_{t-1} + (1-\beta_2)g_t^2$
$\hat{m}_t \leftarrow \dfrac{m_t}{1-\beta_1^t}$
$\hat{v}_t \leftarrow \dfrac{v_t}{1-\beta_2^t}$
$\theta_t \leftarrow \theta_{t-1} - \alpha\,\dfrac{\hat{m}_t}{\sqrt{\hat{v}_t}+\epsilon}$
until θ_t converged

We can see that AdaGrad is a special case of Adam with $\beta_1 = 0$ and β_2 very close to 1. Adam is based on EMA estimates of the moments of the gradient. As the iterations $m_t = \beta_1 m_{t-1} + (1-\beta_1)g_t$ and $v_t = \beta_2 v_{t-1} + (1-\beta_2)g_t^2$ suggest, m_t and v_t are weighted combinations of the previous iterates with g_t and g_t^2, respectively, and hence are adaptive combinations of the "past" and current gradient values. This is one of the most popular optimization methods used in deep learning, and it often outperforms SGD.

2.5 AdaMax

AdaMax [6] is closely related to Adam. We have $m_t = \beta_1 m_{t-1} + (1-\beta_1)g_t$ as before, but instead of v_t, we use a term u_t, which is iteratively given by $u_t = max\{\beta u_{t-1}, |g_t|\}$. This essentially updates the infinity norm of the current and past gradients. Here, we use the update rule $\theta_{t+1} = \theta_t - (\frac{\alpha}{1-\beta_1^t}).\frac{m_t}{u_t}$.

2.6 AdaBelief

AdaBelief [11] is a relatively recent improvement on Adam that bases our step size on the accuracy of our EMA estimate of the gradient. It replaces v_t in the Adam algorithm with $s_t = \beta_2 s_{t-1} + (1-\beta_2)(g_t - m_t)^2 + \epsilon$ and $\hat{s}_t = \frac{s_t}{1-\beta_2^t}$. It then uses the final update $\theta_t = \theta_{t-1} - \alpha \frac{\hat{m}_t}{\sqrt{\hat{s}_t}+\epsilon}$. Considering that $m_t \approx \mathbb{E}[g_t]$, this means that s_t approximates $\mathbb{E}[g_t - \mathbb{E}(g_t)] = Var[g_t]$. Thus, our step size will be directly related to how accurately we can predict g_t, or how much the gradient varies in the neighborhood of θ_t.

3 Experimental Data

To assess the performance of these optimizers, we used each optimizer in a separate neural network that was trained to classify handwritten digits provided by the MNIST data set and mutually exclusive object classes in the CIFAR-10 data set. These problems were deliberately selected because of prior knowledge of the dichotomous performance of Adam on these two problems [10]. The experiment allows all networks to train for 500 epochs on a batch size of 32. This provides a superfluously sufficient environment for all optimizers to converge on a minimizer for the loss function. Both the MNIST and CIFAR-10 data sets were split into training, validation, and test sets of sizes 55000, 5000, and 10000, respectively. All the networks were implemented in Python primarily utilizing the *Pytorch* and *Pytorch Lightning* modules, though a full list of packages used is provided in the "Packages Utilized" section. All models were trained individually on a single NVIDIA GeForce GTX 1660 Ti graphics card.

4 Results and Discussion

We have several guarantees for Adam, AdaGrad, AdaBelief, etc. for convex optimization problems under suitable regularity conditions, including bounded gradients and bounded distance between iterates. We have included a theorem showing this convergence for Adam in Appendix A (Theorem 4.1 and Corollary 4.2) that was proven in [6]. The theorem shows that the regret $R(T)$, the sum of the suboptimality gap at the first T iterates, is bounded by $O(\sqrt{T})$ for Adam. This also holds for AdaGrad and AdaBelief due to their close relationship with Adam. Thus, the average regret for all of these algorithms converges on the order of $O(\frac{1}{\sqrt{T}})$ for convex problems.

However, our loss-minimization problem is not convex since the cross-entropy loss, which includes a softmax activation, is utilized in classification. Chen et al. [1] give a theorem that provides some theoretical guarantees for convergence in non-convex problems for "Adam-type" optimization algorithms, including Adam and AdaGrad. We have provided this theorem in Appendix A (Theorem 2.2), but it requires far stronger conditions than the theorem for convex problems, and these conditions, such as Lipschitz continuity of the gradient of the loss function, are not met in our case. They show that the regret is bounded by $O(\sqrt{T} \cdot \log T)$ under these stronger conditions, which means the average regret converges on the order of $O(\frac{\log T}{\sqrt{T}})$ [1]. Zhuang et al. use this theorem to

demonstrate that under the same conditions, AdaBelief also converges on this order [11].

Figure 1 showcases the training and validation performance of all the optimizers discussed on MNIST. Even though our classification problem does not satisfy the conditions for the convergence of non-convex problems, we see that Adam, AdaGrad, and AdaBelief are all very suitable to predict classes in the MNIST case. This should not come as a surprise, as it is common for Adam and its related derivations to achieve high performance on non-convex problems even when no conditions are met for convergence [1].

Figure 2 is analogous to Figure 1 except all the metrics reflect performance on the CIFAR-10 data set. These plots suggest that the hyperparameters are improperly specified for a setting of 500 maximum training epochs. This is discussed more elaborately in the "Further Work" section.

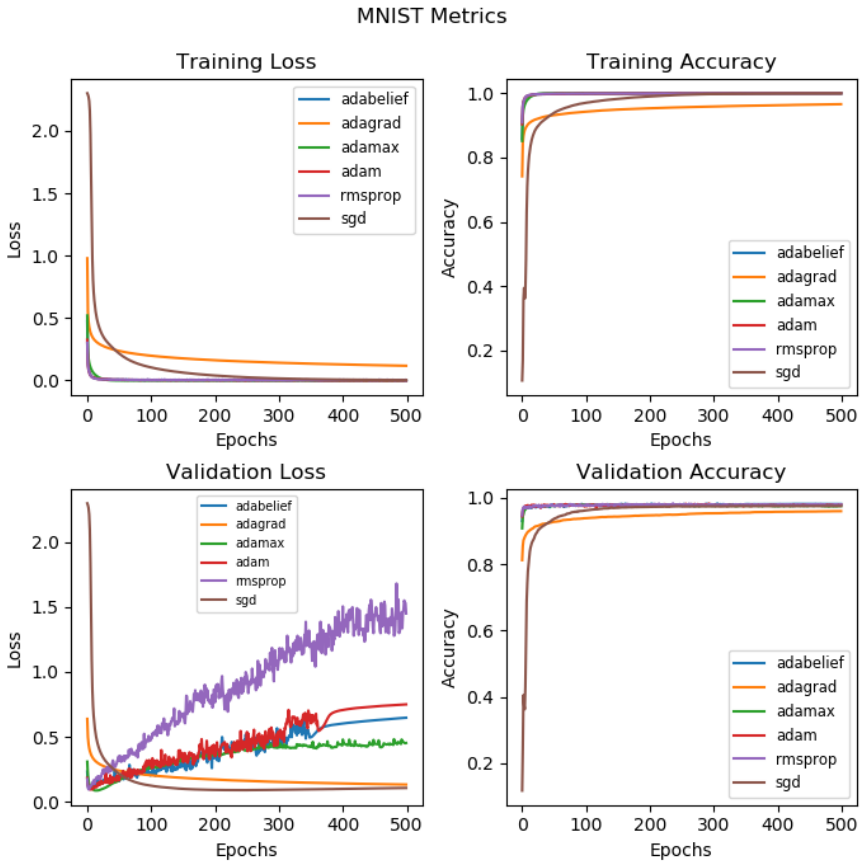

Figure 1: Training and Validation Losses and Accuracy for Various Optimizers on MNIST Neural Net Classification

For the MNIST data, the accuracy achieves near perfect rates almost immediately, with SGD taking a bit longer than the rest of the algorithms, and AdaGrad

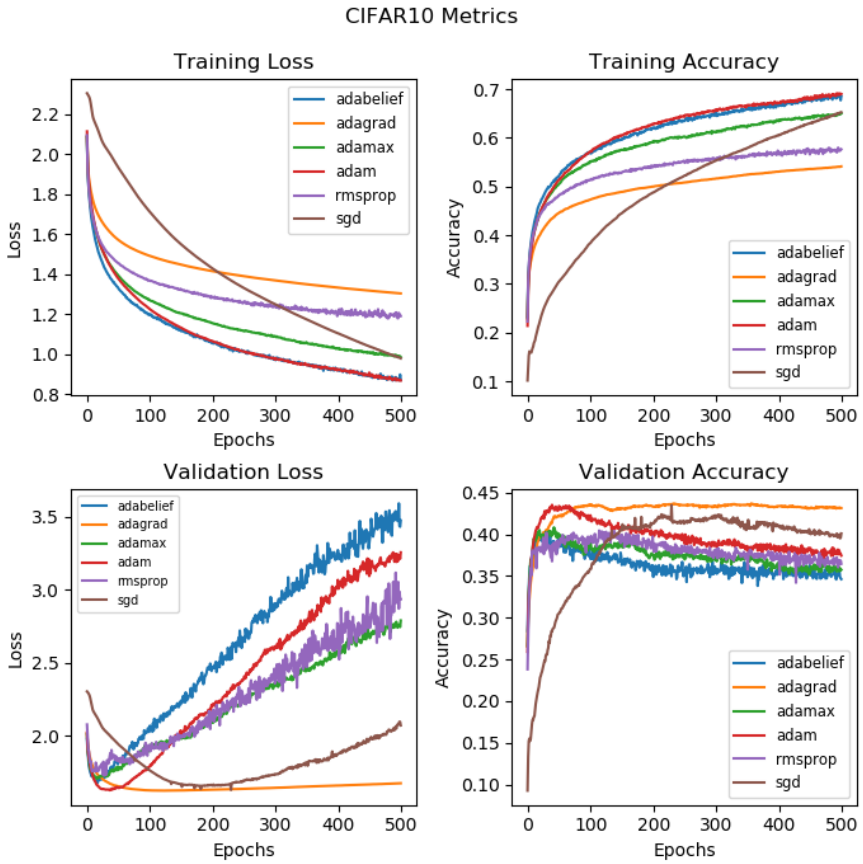

Figure 2: Training and Validation Losses and Accuracy for Various Optimizers on CIFAR-10 Neural Net Classification

not achieving similar rates until after 500 epochs. Figure 1 shows that the loss is in harmony with the accuracy graphs, suggesting that all but SGD and AdaGrad tend to reach an optimal state quickly and further training leads to overfitting. The phenomena at play is that Adam (along with AdaMax, AdaBelief, and RMSProp) suffers from overfitting as the significantly longer training duration follows its quick convergence. This is in line with expected behavior because each epoch has $55000/32 \approx 1718$ updates. Using the result in Theorem 4.2 of Appendix A, the average suboptimality gap under Adam should be less than 0.01 within 10000 iterations, the equivalent of less than a 6 epochs represented on the x-axes in Figure 1. This statement is true regarding the loss, but a decreasing cross-entropy loss need not imply an increase in accuracy. However, a well-defined model should follow this trend in practice and Figures 1 and 2 show our models exhibit this trend.

We would expect AdaBelief to achieve the best performance in terms of convergence rate. Like Adam, AdaBelief also takes a large step in areas where the gradient is small and relatively unchanging, and takes a small step in areas where the gradient is large and variable, such as when the loss function

rapidly approaches a local minimum. However, AdaBelief takes a larger step in areas where the gradient is large and relatively constant since our prediction m_t should be close to g_t, which is a key improvement over Adam [11]. Results on the MNIST data appear to support this hypothesis, but RMSProp, AdaMax, and Adam also showed superior performance here. Yet AdaBelief still generated the highest testing accuracy in the MNIST case, as shown in Table 1. AdaBelief really separated itself in terms of convergence rate on the CIFAR-10 data set, where it was only matched by Adam, but it exhibited dramatic overfitting here and ultimately generated one of the lowest validation and testing accuracies, performing much worse than Adam in this regard.

Figure 3: Random Sample of "8" Digits from MNIST

Figure 4: Random Sample of Horse Images from CIFAR-10

Figures 1 and 2 suggest that optimizers which make updates based on more iterates than only the current one perform significantly worse on the CIFAR-10 data set than the MNIST data set. This result was already provided in [10], but we propose a more intuitive possibility for this result. Figures 3 and 4 showcase some random samples of the MNIST and CIFAR-10 data sets, respectively. There are many similar features between the MNIST samples: same color, same curvature, similar orientation, and similar positioning. This shared information could be a feature that allows momentum-based optimizers like Adam to perform stronger than methods like SGD because their momentum update structure allows for borrowed information across batches with similar input data. The same cannot be said about CIFAR-10. The samples in Figure 4 all have different background colors, horse colors, extraneous objects, angles, positioning, etc. In these situations, optimizers may not help the model learn the new information provided by the diverse new samples because new iterates are a convex combination that include the previous iterates. This weighted update towards previous information makes learning new information that improves classification more challenging. MNIST has similar samples across batches, so it would make sense that the model can learn efficiently, while CIFAR-10s samples have drastically different inputs not only between groups, but within them as well.

	MNIST	CIFAR
AdaBelief	99.10%	40.31%
AdaGrad	96.96%	42.59%
Adam	98.56%	42.66%
AdaMax	98.94%	40.53%
RMSProp	98.84%	39.92%
SGD	98.50%	42.57%

Table 1: Test Accuracy for Vanilla NN of Depth 3 on MNIST and CIFAR Classification Problems

5 Conclusion

As we outlined in Section 2, all these methods have their own pros and cons that contribute to their effectiveness in these two problems. All of the more recent algorithms (those other than AdaGrad and RMSProp) perform well on the MNIST data, and we see that when using $\beta_1 = 0.9$, i.e. β_1 close to 1, we get near perfect validation accuracy for Adam in the MNIST data set. This is in line with many previous works that showcase Adam performing well and converging quickly on the MNIST classifier. We also found evidence of our hypothesis that AdaBelief would achieve slightly better performance than Adam because of how it optimizes step size choices.

For the CIFAR-10 data set, we achieved around 35-45% validation accuracy using vanilla neural networks and similar performance for testing accuracy, suggesting this problem can be better classified using convolutional neural networks with Adam and RMSProp, as detailed in [3]. We found that Adam and AdaBelief achieved the lowest training loss in 500 epochs, but Figure 2 showcases how SGD was on pace to achieve lower training loss with more epochs and was more robust against overfitting, providing evidence that SGD might significantly outperform Adam given more epochs. After 500 epochs though, Adam, SGD, and RMSProp achieved similar maximum testing accuracy, while AdaBelief did not perform nearly as well in this case despite its promising convergence behavior compared to Adam. We propose that Adam performs very well in general on large data sets with similar input data but struggles on data sets with large contrasts, where making new updates based on previous ones hinders the learning process rather than expediting it.

6 Further Work

While some progress has been made to establish Adam's performance relative to other optimization algorithms on arguably the two most common input data sets for neural networks, there is much more to contribute to this work. We found that Adam performs incredibly on MNIST, but struggles on CIFAR-10. While we identified a potential cause for this behavior, it is still conjecture and largely subjective to the idea of what images can be considered "significantly different," but there is not much concrete on this in the literature yet.

Additionally, hyperparameter optimization can greatly increase the performance

of our model. Currently, the hyperparameters are set to their recommended default values, but this does not ensure that our model is efficiently tuned. In fact, SGD should have a much higher learning rate that allows the initial few steps to make faster progress than what is presented in Figure 1. This issue also involves the momentum structure that Adam utilizes (though this does not conform to the usual idea of a hyperparameter). It is unlikely that expanding the number of terms in the momentum structure would change the convergence rate, but some interesting developments could occur with a structure that is fractal or polynomial, and in the future, we could explore that potential.

References

[1] Xiangyi Chen, Sijia Liu, Ruoyu Sun, and Mingyi Hong. On the convergence of a class of adam-type algorithms for non-convex optimization. *CoRR*, abs/1808.02941, 2018.

[2] Yuxin Chen. Stochastic gradient methods, 2019.

[3] Soham De, Anirbit Mukherjee, and Enayat Ullah. Convergence guarantees for rmsprop and adam in non-convex optimization and an empirical comparison to nesterov acceleration, 2018.

[4] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. Technical Report UCB/EECS-2010-24, EECS Department, University of California, Berkeley, Mar 2010.

[5] Geoffrey Hinton, Kevin Swersky, and Nitish Srivastava. Neural networks for machine learning, 2012.

[6] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[7] Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond, 2019.

[8] Sebastian Ruder. An overview of gradient descent optimization algorithms, 2017.

[9] Rachel Ward, Xiaoxia Wu, and Leon Bottou. Adagrad stepsizes: Sharp convergence over nonconvex landscapes, 2021.

[10] Ashia C. Wilson, Rebecca Roelofs, Mitchell Stern, Nathan Srebro, and Benjamin Recht. The marginal value of adaptive gradient methods in machine learning, 2018.

[11] Juntang Zhuang, Tommy Tang, Yifan Ding, Sekhar Tatikonda, Nicha Dvornek, Xenophon Papademetris, and James S. Duncan. Adabelief optimizer: Adapting stepsizes by the belief in observed gradients, 2020.

Packages Utilized

[Package: adabelief_pytorch] J. Zhuang, T. Tang, Y. Ding, S. Tatikonda, N. Dvornek, X. Papademetris, and J. Duncan. Adabelief optimizer: Adapting stepsizes by the belief in observed gradients. *Conference on Neural Information Processing Systems*, 2020.

[Package: matplotlib] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in science & engineering*, 9(3):90–95, 2007.

[Package: numpy] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. Fernández del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. Array programming with NumPy. *Nature*, 585:357–362, 2020. doi:10.1038/s41586-020-2649-2.

[Package: pytorch] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

Appendix A: Theorems

The following theorems, which are referenced in the discussion, show convergence guarantees for Adam originally proven by Kingma and Ba as Theorem 4.1 and Corollary 4.2, respectively, in [6].

Theorem 4.1. *Let* $R(T) = \sum_{t=1}^{T}[f_t(\theta_t) - f_t(\theta^*)]$. *Assume that the function* f_t *has bounded gradients,* $\|\nabla f_t(\theta)\|_2 \leq G, \|\nabla f_t(\theta)\|_\infty \; G_\infty$ *for all* $\theta \in R^d$ *and distance berween any* θ_t *generated by Adam is bounded,* $\|\theta_n - \theta_m\|_2 \leq D$, $\|\theta_m - \theta_n\|_\infty \leq D_\infty$ *for any* $m, n \in \{1, \ldots, T\}$, *and* $\beta_1, \beta_2 \in [0, 1)$ *satisfy* $\frac{\beta_1^2}{\sqrt{\beta_2}} < 1$. *Let* $\alpha_t = \frac{\alpha}{\sqrt{t}}$ *and* $\beta_{1,t} = \beta_1 \lambda^{t-1}, \lambda \in (0, 1)$. *Adam achieves the following guarantee, for all* $T \geq 1$.

$$R(T) \leq \frac{D^2}{2\alpha(1-\beta_1)} \sum_{i=1}^{d} \sqrt{T\hat{v}_{T,i}} + \frac{\alpha(1+\beta_1)G_\infty}{(1-\beta_1)\sqrt{1-\beta_2}(1-\gamma)^2} \sum_{i=1}^{d} \|g_{1:T,i}\|_2 + \sum_{i=1}^{d} \frac{D_\infty^2 G_\infty \sqrt{1-\beta_2}}{2\alpha(1-\beta_1)(1-\lambda)^2}$$

Corollary 4.2. *Assume that the function* f_t *has bounded gradients,* $\|\nabla f_t(\theta)\|_2 \leq G, \|\nabla f_t(\theta)\|_\infty \leq G_\infty$ *for all* $\theta \in R^d$ *and distance between any* θ_t *generated by Adam is bounded,* $\|\theta_n - \theta_m\|_2 \leq D$, $\|\theta_m - \theta_n\|_\infty \leq D_\infty$ *for any* $m, n \in \{1, \ldots, T\}$. *Adam achieves the following guarantee, for all* $T \geq 1$

$$\frac{R(T)}{T} = O\left(\frac{1}{\sqrt{T}}\right)$$

The following theorem, which is also referenced in the discussion, shows a convergence guarantee for AdaBelief in non-convex problems. The theorem appears as Theorem 2.2 in [11] but was originally proven for classes of optimization algorithms closely related to Adam by Chen et al. in [1].

Theorem 2.2. *(Convergence for non-convex stochastic optimization) Under the assumptions:*

- f *is differentiable;* $\|\nabla f(x) - \nabla f(y)\|] \leq L\|x - y\|, \forall x, y : f$ *is also lower bounded.*
- *The noisy gradient is unbiased, and has independent noise, i.e.* $g_t = \nabla f\left(\theta_t\right) + \zeta_t, \mathrm{E}\zeta_t = 0, \zeta_t \perp \zeta_j, \forall t, j \in \mathbb{N}, t \neq j$
- *At step* t, *the algorithm can access a bounded noisy gradient, and the true gradient is also bounded. i.e.* $\|\nabla f\left(\theta_t\right)\| I \leq H, \|g_t\| \leq H, \forall t > 1$.

Assume $\min_{j \in [d]} \left(s_1\right)_j \geq c > 0$, *noise in gradient has bounded variance,* $\mathrm{Var}\left(g_t\right) = \sigma_t^2 \leq \sigma^2, s_t \leq s_{t+1}, \forall t \in \mathbb{N}$, *then the proposed algorithm satisfies:*

$$\min_{t \in [T]} \mathbb{E}||\nabla f\left(\theta_t\right)\|^2 \leq \frac{H}{\sqrt{T\alpha}} \left[\frac{C_1 \alpha^2 \left(H^2 + \sigma^2\right)\left(1 + \log T\right)}{\epsilon} + C_2 \frac{d\alpha}{\sqrt{c}} + C_3 \frac{d\alpha^2}{c} + C_4\right]$$

as in [1] C_1, C_2, C_3 *are constants independent of* d *and* T, *and* C_4 *is a constant independent of* T.

Appendix B: Supplementary Materials

The code, data, and figures used throughout this paper as well as instructions to reproduce these results can be found at `https://github.com/roromaniac/project_ceo`.