Meeting Adam: A Reproduction Study of the Original Adam Optimizer Paper

Anonymous authors
Paper under double-blind review

Abstract

Adam is one of the most widely used optimizers for deep learning. Kingma & Ba (2015) reported strong performance across logistic regression, fully connected neural networks, and convolutional neural networks, comparing Adam with SGD with Nesterov momentum, AdaGrad, RMSProp, and AdaDelta. In this report I reproduce the experiments in Section 6 of the original paper using some simplified modern implementations. I implement Adam from scratch using NumPy and I compare the original claims to my reproduced learning curves on MNIST logistic regression, IMDB bag-of-words logistic regression, a multilayer neural network with dropout on MNIST, and a CIFAR-10 convolutional network. Overall I find that the qualitative conclusions of the paper largely hold: Adam matches or outperforms AdaGrad on sparse problems and competes with tuned SGD with Nesterov momentum on deep networks. However, there are notable quantitative differences, particularly for the CIFAR-10 convolutional neural network: in several settings Adam struggles to outperform SGD with Nesterov momentum, despite testing a wide range of learning rates, L_2 regularization strengths, and architectural variants. I conclude with a discussion of likely causes for these discrepancies and practical lessons for using Adam in modern setups.

1 Introduction

The Adam optimizer (Kingma & Ba, 2015) has become a default choice for training deep learning models due to its combination of adaptive learning rates (RMSProp) and momentum. Given its influence, it is important to assess how robust the original empirical findings are to reimplementation and moderate changes in the experimental environment.

This report presents a similar-scale reproduction study of the key experiments in Sections 6 of Kingma & Ba (2015). The focus is on comparing learning dynamics rather than achieving identical results. I reimplement the original models using NumPy and opting for PyTorch implementations where reasonable, tune the same family of optimizers, and compare the resulting training curves to those in the paper while matching the original structure and parameters where possible.

I reproduce the following sections from the original paper: (6.1) Logistic regression on MNIST and IMDB bag-of-words features. (6.2) A two-hidden-layer fully connected network with dropout on MNIST. (6.3) A convolutional neural network on CIFAR-10. (6.4) Bias-correction experiments.

2 Original Experiments and Claims

2.1 Experiment 6.1: Logistic Regression

Section 6.1 studies L2-regularized multi-class logistic regression on MNIST and binary logistic regression on IMDB bag-of-words features. The objective is convex, making it suitable for comparing optimizers without local minima complications. The authors apply a decaying stepsize $\alpha_t = \alpha/\sqrt{t}$ and compare AdaGrad, SGD with Nesterov momentum, and Adam using minibatches of size 128.

Their reported findings are:

- On dense MNIST pixels, Adam and SGD with Nesterov momentum show similar convergence and both significantly outperform AdaGrad.
- On sparse IMDB bag-of-words features (with 10 000 dimensions and 50% dropout on the input), AdaGrad, RMSProp (i.e. Adam without momentum) and Adam significantly outperform SGD with Nesterov momentum. Adam converges about as fast as AdaGrad, consistent with its design to inherit AdaGrad's benefits on sparse data.

2.2 Experiment 6.2: Multilayer Neural Network on MNIST

Section 6.2 considers a fully connected neural network with two hidden layers of 1000 ReLU units each, trained on MNIST with minibatches of size 128. The authors first compare Adam against SFO, a quasi-Newton method, on a deterministic network with L2 weight decay, and report that Adam converges faster both in iterations and wall-clock time. They then add dropout regularization and compare first-order methods (AdaGrad, RMSProp, SGD with Nesterov momentum, AdaDelta, Adam), finding that Adam shows the best convergence among these methods.

2.3 Experiment 6.3: Convolutional Neural Networks on CIFAR-10

Section 6.3 evaluates Adam on a convolutional neural network with the architecture c64-c64-c128-1000: three stages of 5×5 convolution and 3×3 max pooling (stride 2) followed by a fully connected layer with 1000 ReLU units. Dropout is applied to the input and fully connected layers, and the minibatch size is 128. The key qualitative findings are:

- In the first few epochs, Adam and AdaGrad rapidly reduce the training cost.
- Over 45 epochs, AdaGrad eventually stalls while Adam and SGD with Nesterov momentum continue to make progress and achieve substantially lower training cost.
- Adam's final performance is only marginally better than that of SGD with momentum on this task, but it has the practical advantage of automatically adapting learning rates across layers.

2.4 Experiment 6.4: Bias-Correction Term

Section 6.4 investigates Adam's bias-correction terms for the first and second moments. The authors compare Adam with and without bias correction (the latter being equivalent to a momentum version of RMSProp) on a variational autoencoder (VAE) with a single 500-unit softplus hidden layer and a 50-dimensional Gaussian latent variable. They vary $\beta_1 \in \{0, 0.9\}$, $\beta_2 \in \{0.99, 0.999, 0.9999\}$, and $\log_{10}(\alpha) \in \{-5, \ldots, -1\}$, and plot training loss after 10 and 100 epochs. The main claim is that for large β_2 (needed when gradients are sparse), omitting bias correction leads to unstable or divergent behavior, while Adam with bias correction remains stable and typically performs at least as well.

3 Reproduction Specifics and Results

All experiments with the exception of section 6.1 were implemented in PyTorch using a single GPU for training. I attempted to match the original architectures, batch sizes, and choice of optimizers as closely as possible, while keeping the code simple and using built-in optimizer implementations. However, some sections did not have full details such as learning rates, dropout amounts, etc.

For Adam I used the default hyperparameters recommended by Kingma & Ba (2015): $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, however the learning rate was adjusted per experiment. For the other optimizers I performed a small grid search over learning rates selected the best curve for plotting.

3.1 Experiment 6.1: Logistic Regression

MNIST Setup. I use the standard MNIST split (60k training, 10k test). Images are reshaped into 784 dimensional vectors and scaled to $[0, 1]$. The model is a single linear layer with softmax, trained with L_2 weight decay. I train for 45 passes over the training set (matching the horizontal axis in the original figure). Figure 1 shows the reproduced training curves for logistic regression on MNIST and IMDB. All methods converge smoothly. As in the original paper, Adam and SGD with Nesterov momentum both converge faster than AdaGrad. However, my reproduction shows a more uniform difference between the three optimizers, Adam and SGD Nesterov show similar behavior to the original paper when accounting for scaling differences, but notably AdaGrad converges faster as compared to the original paper. This suggests that with my particular learning-rate parameters, the learning rate for AdaGrad may have been too low.

IMDB Bag-of-Words Setup. Following the paper, I construct 10,000-dimensional bag-of-words features using the most frequent tokens in the IMDB movie review dataset. The logistic regression model predicts positive vs. negative sentiment. During training I apply 50% dropout noise to the input BoW vector. On IMDB, AdaGrad, RMSProp (with dropout), and Adam all rapidly reduce the training cost and end up with similar final values. SGD with Nesterov momentum, however, exhibits significant instability with large spikes and a much higher overall cost. Qualitatively this agrees with the claim that Adam matches AdaGrad on sparse features and that SGD struggles. Quantitatively, Adam is slightly ahead of RMSProp in my curves, while the original figure suggests almost identical performance between the two methods with further divergence in later iterations.

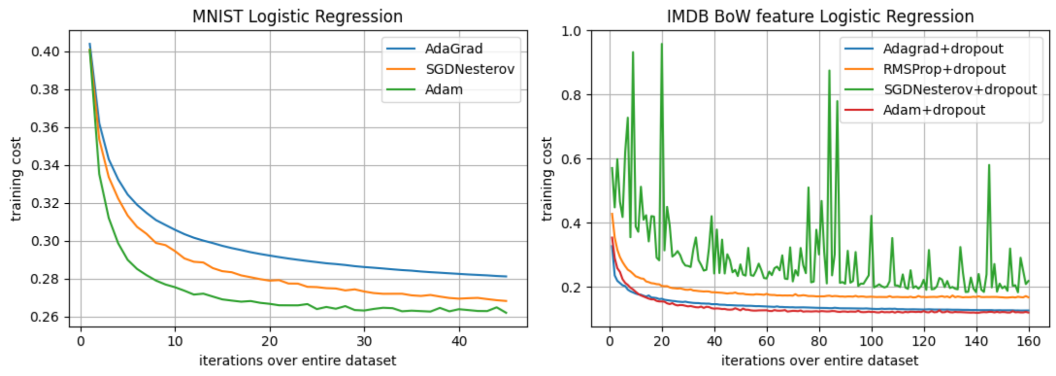


Figure 1: Reproduction of the logistic regression experiments (Section 6.1). Left: MNIST multi-class logistic regression. Right: IMDB bag-of-words logistic regression with 50% dropout on the input.

3.2 Experiment 6.2: Multilayer Neural Network on MNIST

The reproduced MLP uses two fully connected hidden layers with 1000 ReLU units each and a 10-way softmax output, trained on MNIST with minibatch size 128 and 50% dropout on the hidden units. I reproduce only the stochastic, dropout regularized setting (the left panel of Figure 2 in the original paper); the deterministic network and SFO baseline are omitted for simplicity.

Figure 2 shows my reproduction of the dropout regularized MLP experiment.

The qualitative ranking of optimizers matches the original report: Adam achieves the lowest training cost and converges faster than AdaGrad, RMSProp, AdaDelta, and SGD with

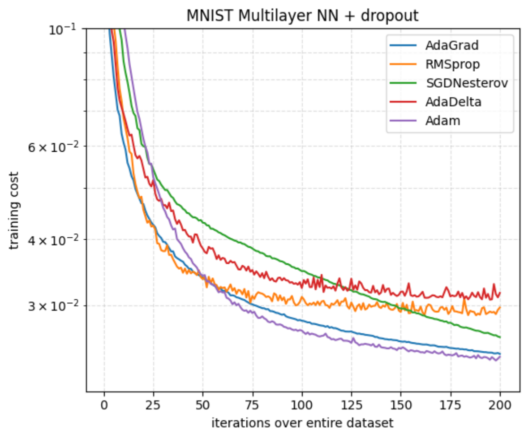


Figure 2: Reproduction of the MNIST multilayer neural network with dropout (Section 6.2). Training cost versus passes over the dataset.

Nesterov momentum. The ordering of the baselines is also similar (RMSProp and AdaDelta form a middle group, with AdaGrad and SGD somewhat worse), though Adam and AdaGrad are closer. This may reflect differences in regularization strength, weight initialization, or the exact learning rate choices (specifically LR has a big effect here).

3.3 Experiment 6.3: Convolutional Neural Network on CIFAR-10

For CIFAR-10 I implement the c64-c64-c128-1000 architecture described above, using 5×5 convolutions, 3×3 max pooling with stride 2, ReLU activations, and a fully connected layer of 1000 units before the 10-way softmax. Input images are normalized (with known values for the mean and standard deviation). Dropout is applied to the input and fully connected layers. I train with minibatch size 128 for 45 epochs, as in the original work. Figure 3 presents the reproduced CIFAR-10 ConvNet curves for the first three epochs and for the full 45 epochs.

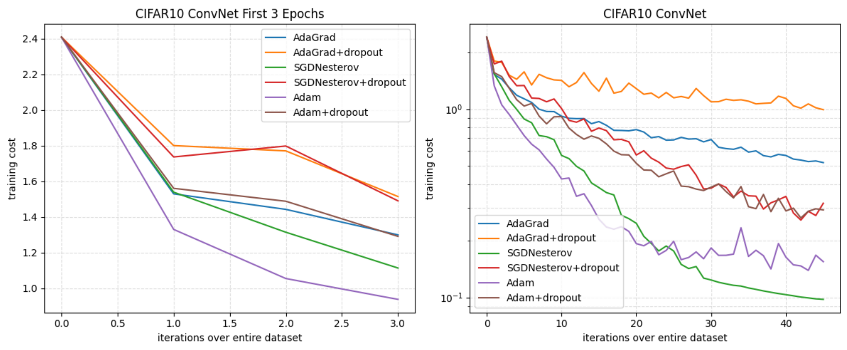


Figure 3: Reproduction of the CIFAR-10 ConvNet experiments (Section 6.3). Left: training cost over the first three epochs. Right: training cost over 45 epochs (log scale).

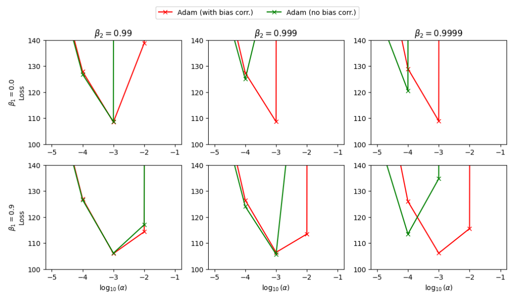
Early training (first three epochs). As in the original figure, Adam quickly reduces the training cost. However other optimizers show a different ordering likely due to the choice of random initialization - notably SGD Nesterov follows Adam closely.

Full training (45 epochs). Over 45 epochs, my results vary from the original paper, in regards to Adam's performance. All pairings of optimizers are the same; AdaGrad with and without dropout are similar but with more divergence, Adam+dropout is paired with

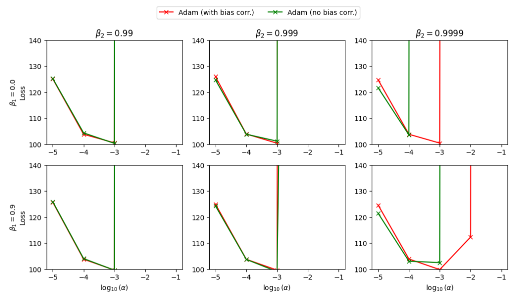
SGDNesterov+dropout, and Adam is paired with SGDNesterov with divergence around epoch 30. My results are considerably noisier, and show that Adam reaches a cost floor earlier than SGD Nesterov, both of which fail to reach a cost of around 10^{-3}. Learning rates in the range $[0.00005 - 0.01]$ were tried as well as delayed weight decay, different beta values, and even larger network sizes - but no configurations could reproduce the original results. In some configurations, which were different from the original paper, SGD Nesterov achieved a cost of around 10^{-3} but Adam did not, this may suggest issues with my implementation. The original graph also suggests some type of smoothing or averaging was applied for appearance.

3.4 Experiment 6.4: Bias-Correction Term

To reproduce Section 6.4, I train a VAE with the same architecture as in Kingma & Ba (2015): a single hidden layer with 500 softplus units and a 50-dimensional spherical Gaussian latent variable. I use binarized MNIST as the dataset and optimized the standard variational lower bound with minibatch size 100.



(a) After 10 epochs



(b) After 100 epochs

Figure 4: Effect of Adam's bias-correction terms when training a VAE on binarized MNIST. Each panel plots the final loss versus $\log_{10}(\alpha)$ for a fixed (β_1, β_2) combination. Red: Adam with bias correction. Green: Adam without bias correction (RMSProp-like variant).

10 epochs. After 10 epochs (Figure 4a), Adam with and without bias correction behave similarly for most hyperparameter settings. Both versions achieve their best loss around $\log_{10}(\alpha) \approx -3$, with only mild degradation at more extreme stepsizes. For smaller β_2 values (0.99), removing bias correction has almost no visible effect. For $\beta_2 = 0.9999$ there are already hints of instability in the uncorrected variant for the largest stepsize, but overall the curves still roughly coincide.

100 epochs. After 100 epochs (Figure 4b), the differences become pronounced, especially for $\beta_2 \in \{0.999, 0.9999\}$. With bias correction, Adam consistently finds a well-shaped minimum around $\log_{10}(\alpha) \approx -3$, and the loss remains within a narrow band across neighboring stepsizes. Without bias correction, several settings either diverge (loss at the top of the plot) or produce worse minima, particularly for $\beta_2 = 0.9999$ and larger stepsizes. The pattern matches the original claim that when β_2 is close to 1 the initialization bias in the second-moment estimate leads to overly aggressive steps and training instabilities.

Overall, the reproduction agrees with the qualitative conclusion of Section 6.4: bias correction is most important when β_2 is close to one and training has progressed long enough that the early instabilities can compound. In no configuration did the uncorrected variant outperform Adam with bias correction by a meaningful margin.

3.5 Summary of Agreement and Discrepancies

Experiment	Summary
(6.1) MNIST log. reg.	Original: Adam $\approx$ Nesterov SGD $>$ AdaGrad Reproduced: Yes (Though AdaGrad is closer)
(6.1) IMDB BoW log. reg.	Orig: Adam $\approx$ AdaGrad $\approx$ RMSProp $>$ SGD Nesterov Rep: Yes (Adam slightly better, SGD unstable)
(6.2) MNIST MLP + dropout	Orig: Adam best among first-order methods Rep: Yes (same ranking, but smaller gaps)
(6.3) CIFAR-10 ConvNet	Orig: Adam & SGD $>$ Adam+d & SDG+d $>$ AdaGrad $\pm$ d Rep: Moderate (same pairings, different costs, and noisier)
(6.4) VAE bias correction	Orig: Bias-corrected Adam $\geq$ uncorrected RMSProp-like variant, especially for large β_2 Rep: Yes (uncorrected often unstable or worse)

Table 1: Qualitative comparison between the original experimental claims and the reproduced results.

4 Discussion

Overall the reproduction supports the main narrative of Kingma & Ba (2015): Adam is competitive with tuned SGD with Nesterov momentum on dense deep networks and inherits AdaGrad's strengths on sparse features. Where discrepancies appear, especially in (6.3), the reproduction still generally follows the original claims rather than contradict the original conclusions.

There are several plausible explanations for the quantitative differences:

- I used a relatively small (manual) learning-rate grid and may not have found the exact best setting for each baseline, whereas the original paper reports a "dense grid" search.
- Modern frameworks may differ in default initializations, implementations, and/or other details.
- All curves shown here are from single runs. Averaging over multiple random seeds would give a more robust estimate of optimizer performance.
- The original paper does not report all hyperparameters or details (E.g. L_2 reg. is used in 6.1, 6.2, but 6.3?).

5 Conclusion

This reproduction study confirms that the central empirical claims of Adam: A Method for Stochastic Optimization are robust to a modern reimplementation. Adam remains a strong default choice, particularly on problems with sparse gradients or when per-layer learning rate tuning is arduous.

References

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