Hyperparameter Sensitivity Analysis for Adam Optimization on Deep CNNs

Abhishek Madan
University of Toronto

Fei Wu
University of Toronto

Abstract

Recent work in deep learning uses the Adam optimization procedure as a more effective alternative to stochastic gradient descent, which uses information from previous derivatives to rescale descent direction components and improve behaviour in high-curvature regions of the cost landscape. Although quite successful, it is often viewed as a black box; anecdotal advice for using Adam often amounts to using the parameters given in the paper, with little general wisdom for tuning them to potentially improve performance. To improve our understanding of this algorithm on more complex models, we implemented a VGG and ResNet network, two families of deep CNN-based image classification models, and analyzed their behaviour using different Adam hyperparameter values. We find that optimization with Adam is stable with respect to small perturbations in hyperparameters, and the default parameters all achieve good performance, which further reinforces the choice of default parameters. Furthermore, the moment decay parameters unique to Adam are, surprisingly, fairly insensitive, and a wide range of values result in efficient training.

1 Introduction

A good stochastic optimization algorithm is crucial for high performance in deep learning, both to converge training in fewer epochs and to achieve models with lower cost within a fixed training budget. Recently, Adam [Kingma and Ba, 2017] has enjoyed frequent use as a first-order method which uses first- and second-order moments of the gradient to improve convergence in regions with sparse gradients and high curvature. However, unlike its simpler relatives like stochastic gradient descent (SGD), its hyperparameters are often not tuned, and practitioners simply choose to use the values presented in the original paper. Although the hyperparameters have simple interpretations, it is difficult to truly understand their effects on optimizing highly non-convex cost functions, and how small changes in the hyperparameters can affect convergence (i.e., sensitivity analysis). The authors chose those parameters using a dense grid search, but did not include the results in the paper.

To bridge this gap in understanding, we will analyze Adam and its effect on two image recognition models: VGG [Simonyan and Zisserman, 2015] and ResNet [He et al., 2015]. These models have many more layers than the convolutional neural network investigated in the original Adam paper, and provide an opportunity to investigate Adam's behaviour on two successful CNN architectures. For each model, we analyze a variety of hyperparameter settings for Adam, and view their training loss after a few epochs of training. This information allows us to infer the sensitivity of training speed on Adam's hyperparameters. In summary, we find that training loss is actually relatively insensitive to small perturbations in hyperparameters, and the default hyperparameters also perform well for deeper CNNs. Furthermore, there is a wide range of values for all hyperparameters that also perform well for these models.

Preprint. Under review.

2 Related Works

Adam [Kingma and Ba, 2017] is a stochastic gradient decent algorithm which combines RMSprop and AdaGrad [Duchi et al., 2011]. It utilizes an exponential moving average of the first and second moments of the gradients. Variants of Adam like AdaBound [Luo et al., 2019] improve generalization through dynamic learning rates. For the purposes of the course project, we will focus on Adam rather than more complex follow-up methods.

VGG [Simonyan and Zisserman, 2015] showed that small convolution kernels and several layers could achieve comparable results to models with larger kernels and fewer layers while using far fewer weights. It was a highly successful image recognition model, winning first place in ILSVRC 2014. More recent models like ResNet [He et al., 2015] achieve superior performance, but VGG provides a good model to analyze for the course project, where the individual pieces of the model are straightforward.

ResNet [He et al., 2015] is motivated by the vanishing gradient problem. It allows easier deeper neural network training and achieved 1st place in ILSVRC & COCO 2015 competitions, and they demonstrated superior performance to VGG. A ResNet model is more complex than a VGG model and should give us further insight into the effect of Adam's hyperparameters.

3 Method

3.1 Overview of Adam

We are interested in the effects of Adam's hyperparameters on training loss over time. For ease of reference, we provide a reproduction of the algorithm below, and then discuss its behaviour.

Algorithm 1: Adam optimization algorithm, reproduced from [Kingma and Ba, 2017].

```
Input  : Learning rate $\alpha$; decay rates $\beta_1$, $\beta_2$; divide-by-zero offset $\epsilon$; stochastic objective
         function $f(\theta)$; initial estimate $\theta_0$
Output : Optimal parameter estimate $\theta^*$
1 $m_0 \leftarrow 0$, $v_0 \leftarrow 0$, $t \leftarrow 0$
2 while $\theta_t$ not converged do
3     $t \leftarrow t + 1$
4     $g_t \leftarrow \nabla_{\theta_{t-1}} f(\theta_{t-1})$
5     $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$
6     $v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$
7     $\theta_t \leftarrow \theta_{t-1} - \alpha \frac{m_t/(1-\beta_1^t)}{\sqrt{v_t/(1-\beta_2^t)}+\epsilon}$
8 end
9 return $\theta_t$
```

As seen from the pseudocode, the learning rate α controls the step size taken at each iteration, while β_1 and β_2 control the decay rates of the exponential moving averages m_t and v_t, respectively, where the former is an estimate of the first moment and the latter is an estimate of the (uncentered) second moment. There is also a parameter ϵ which is mainly used to avoid divide-by-zero errors, but we note that a sufficiently high ϵ essentially eliminates the effect of the second moment and causes Adam to resemble SGD with momentum.

(The following discussion will apply to both first and second moment estimates, and to indicate this, we drop the subscript on β.) There are also scale factors of the form $1 - \beta^t$ on both moment estimates, which are used to eliminate estimation bias. To understand why we need these correction factors, consider an objective function with sparse gradients, with entries that are mostly zero: if β is too low, then the moment estimate will be very sensitive to slight changes in the gradient, but if β is too high, then the estimate is not responsive enough to new information and may take poor steps. Intuitively, the bias correction scale factors fix this by amplifying new data at earlier iterations to make rapid progress, and lessening this effect in later iterations where we are more likely to be near a local minimum and do not need to rapidly change descent direction.

3.2 Network Architectures

To test Adam, we implement two similar CNNs: VGG and ResNet, on the CIFAR-10 image classification dataset. Rather than specific network architectures, both of these are actually families of architectures based on using small kernels (VGG) and using small kernels and skip connections (ResNet). Since the ResNet paper is a follow-up to the VGG paper, we implement two architectures described in the ResNet paper: one is what the authors call a "plain network" that is essentially a VGG network, and the other is the 34-layer architecture shown in Figure 4 of [He et al., 2015]. The plain/VGG network is described in Section 4.2 of [He et al., 2015], where convolution operations are split into 3 sets of layers containing 6 layers each, where transitions between layer sets are marked by pooling operations implemented with strided convolutions in the last layer of each set. This architecture was designed for CIFAR-10, in contrast to the ResNet architecture which was designed for ImageNet (though can be easily adapted to work for CIFAR-10). Both networks are much deeper than the 4-layer CNN tested in [Kingma and Ba, 2017], so the results in that paper do not necessarily predict the behaviour we will see in the experiments we conduct.

3.3 Experimental Setup

Using these two CNN architectures, we will vary the four described Adam hyperparameters and test the sensitivity of these parameters on training loss (cross-entropy loss). We will perform two sets of tests: relatively small perturbations centered around the default parameter values, and order-of-magnitude changes where parameters are varied (approximately) logarithmically. The former tests help us understand the sensitivity of the parameters and the usefulness of these default parameters on deeper networks, and the latter tests allow us to explore the parameter space in this deep CNN setting. Due to time constraints, we do not perform a grid search, and only modify one parameter at a time during tests. We will perform only order-of-magnitude tests on ϵ since the effect of small perturbations will likely be minimal, since it is already so small.

We also examine the effects of hyperparameters on test loss and test accuracy in Appendix C.

4 Results

All the code is written in Python using the PyTorch library, and has been included with the submission. In particular, we use the PyTorch implementation of Adam, and implement the two neural networks using PyTorch's neural network building API and automatic differentiation capabilities. For simplicity, we do not use dropout regularization, and we do not augment the training dataset in any way (e.g., horizontal flips, translation+cropping). In order to make results consistent between experiments and to accommodate all the experiments being run in limited time in Google Colab, we fix the training batch size to be 100 and training epoch count at 15. Also due to time constraints, each experiment is only run once, though this does lead to some noisy results, particularly for ResNet.

In the perturbation tests, we see little change in training loss curves across all hyperparameters and both networks — see Appendix B for loss plots. This suggests that Adam is "stable" in the sense that it will not exhibit pathologically poor performance due to slight changes in the hyperparameters, which makes sense because all the selected values are "nice" (few significant digits, only digits are 1 and 9) and were selected manually. If they were highly sensitive, then such a procedure would not work and a dense grid search would be needed to find good parameter values. Therefore, from an educational standpoint, the wide adoption of this algorithm without hyperparameter tuning seems much more understandable.

The training loss curves for all order-of-magnitude experiments are given in Figure 1. By inspection, we see that the most sensitive parameter is α, with wider variations in loss curves than for the other hyperparameters. The learning rate is known to be important in simpler algorithms like SGD, so its relative sensitivity is not surprising; however, it is interesting to see that, for VGG, the default 0.001 is actually the best of the 5 candidate values, where changes in order of magnitude correspond to higher loss plots, reinforcing the default choice. For ResNet, we also see the default perform well, but smaller learning rates also achieve competitive performance, with 10^{-5} achieving the best performance, though the sharp jumps in the curve seem to indicate this is due to noise. Nevertheless, the loss curve using 0.0001 is very similar to 0.001, which suggests that ResNet can be optimized with a smaller learning rate. [Kingma and Ba, 2017] describe α as a trust region radius due to the

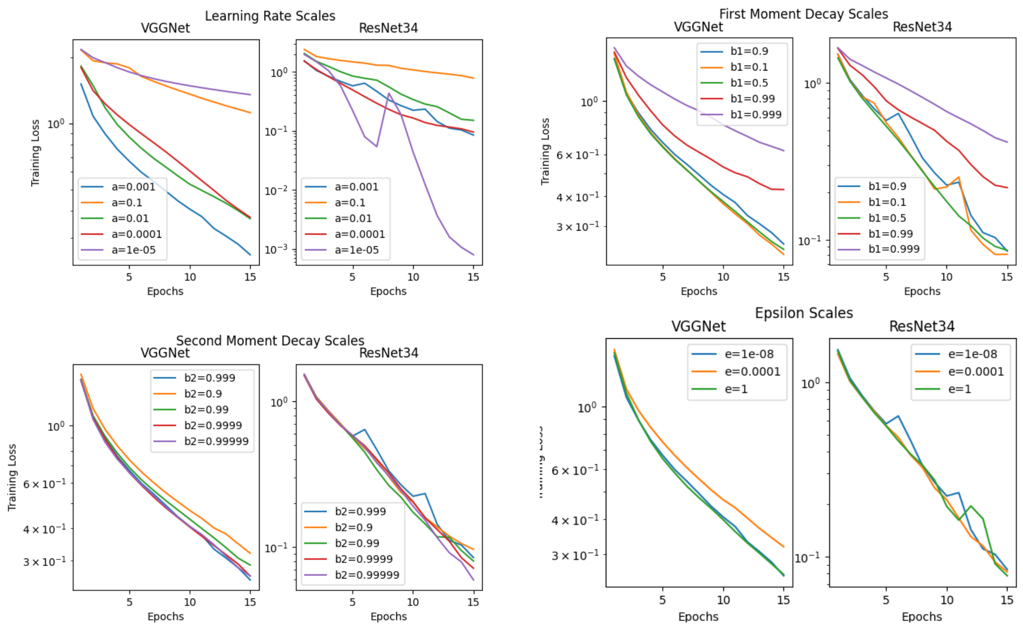

Figure 1: Training loss curves for experiments on α (top left), β_1 (top right), β_2 (bottom left), and ϵ (bottom right). Y axes are logarithmic.

rescaling done by the moment estimates, so a smaller usable learning rate implies that ResNets have a smaller trust region, which typically occurs close to a stationary point. This further reinforces the hypothesis made by [He et al., 2015] that identity mappings are near a local minimum.

The next parameter with some loss curve variation is the first moment decay β_1, however the plots only show sensitivity when β_1 is too high. We see comparable performance between the lowest decay of 0.1 and the default of 0.9, which suggests there is a wide range of good values for β_1. Since β_1 can be interpreted as a momentum decay, this wide range implies that the decay rate matters much less than (a) incorporating at least some momentum to avoid getting stuck in high-curvature regions of the objective landscape, and (b) using enough new gradient information to make useful progress.

The second moment rescaling, controlled by β_2 and ϵ, is shown to be relatively insensitive to both parameters, though we do see that higher β_2 tends to perform slightly better, which was the recommendation given in [Kingma and Ba, 2017]. However, the insensitivity to ϵ is unexpected, and due to its interpretation as adjusting between RMSprop-like second moment scaling and SGD with momentum, it implies that both ends of the spectrum are equally effective at training these two networks (which can in fact be seen in the original Adam paper [Kingma and Ba, 2017]). Note that we did not estimate gradient magnitude when selecting ϵ, but due to high network depth, we expect small gradients (or gradients close to 1 for ResNet), so a maximum tested value of 1 is reasonable.

5 Conclusion

In conclusion, we find that the default Adam parameters indeed work well for deep CNNs using VGG and ResNet architectures, despite not being tested in the original paper. In the process, we also reinforce a hypothesis from the original ResNet paper [He et al., 2015] that close-to-identity mappings are optimal for CNNs. However, interestingly, we see that the less intuitive moment decay parameters are quite insensitive, and a wide range of parameters result in efficient training. Therefore, the common trend to use the default values for these parameters can be interpreted as being due to a lack of necessity to fine-tune them, instead of a necessity to leave them as-is to use a highly sensitive algorithm. In fact, the ϵ results suggest that other algorithms like SGD with momentum can be used instead to train CNNs, suggesting a much wider space of usable algorithms than students like us would initially believe.

References

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

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(61):2121–2159, 2011. URL `http://jmlr.org/papers/v12/duchi11a.html`.

Liangchen Luo, Yuanhao Xiong, Yan Liu, and Xu Sun. Adaptive gradient methods with dynamic bound of learning rate, 2019.

A Contributions

Abhishek Madan set up the Jupyter notebook used for experiments, implemented the VGG model, ran experiments, and wrote the report. **Fei Wu** implemented the ResNet model and ran experiments.

B Hyperparameter Perturbation Results

The hyperparameter perturbation training loss curves are shown in Figure 2. As shown, the loss curves are all close together, which indicates the hyperparameters are insensitive to slight changes, which is a desirable property for a widely used optimization algorithm.

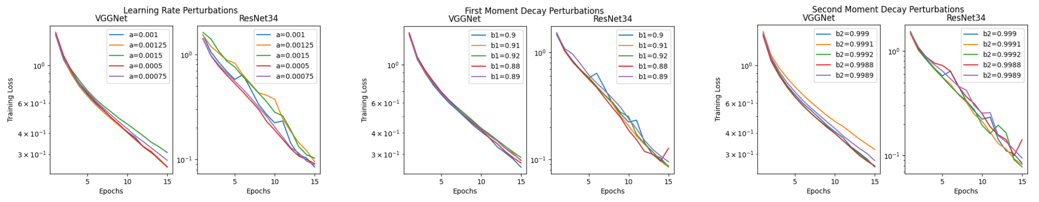

Figure 2: Training loss curves for perturbation experiments on α (left), β_1 (middle), and β_2 (right). Y axes are logarithmic.

C Validation Loss Experimental Results

Validation loss curves for the order-of-magnitude experiments are given in Figure 3, and the validation accuracy curves are given in Figure 4. As expected, these curves are much noisier than the training loss curves, since the optimizer does not see these values during training. Interestingly, the validation loss curves all tend to form a flat or V shape, though as shown in Figure 4, accuracy does increase over time and plateau at roughly 75%. These results are likely due to the network converging towards unimodal classification distributions for most input images, though with some misclassifications in the validation set (the CIFAR-10 test set).

The learning rate validation results are mostly consistent with the training loss results, in that the default rate performs well on both models, and very low and very high learning rates perform relatively poorly for VGG. However, we also see that low learning rates for ResNet do not result in high accuracy on the validation dataset, despite comparable training losses. The validation loss for 0.0001 is slightly worse than the default 0.001, and validation accuracy is significantly worse, only achieving roughly 65% accuracy compared to the default 75%. In a more extreme example, the 10^{-5}-trained ResNet model performs much worse, with the highest validation loss and lowest validation accuracy, despite having much lower training loss than other values. This complicates the conclusion in the main paper somewhat about the identity mapping being close to optimal, as

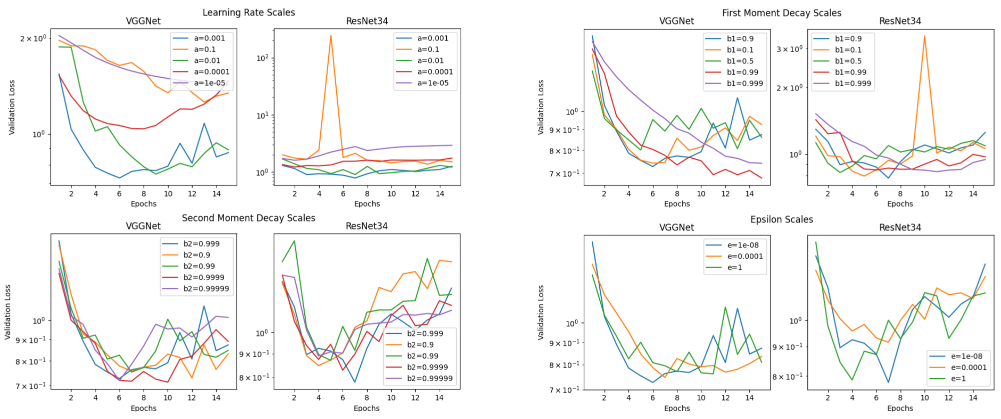

Figure 3: Validation loss curves for experiments on α (top left), β_1 (top right), β_2 (bottom left), and ϵ (bottom right). Y axes are logarithmic.

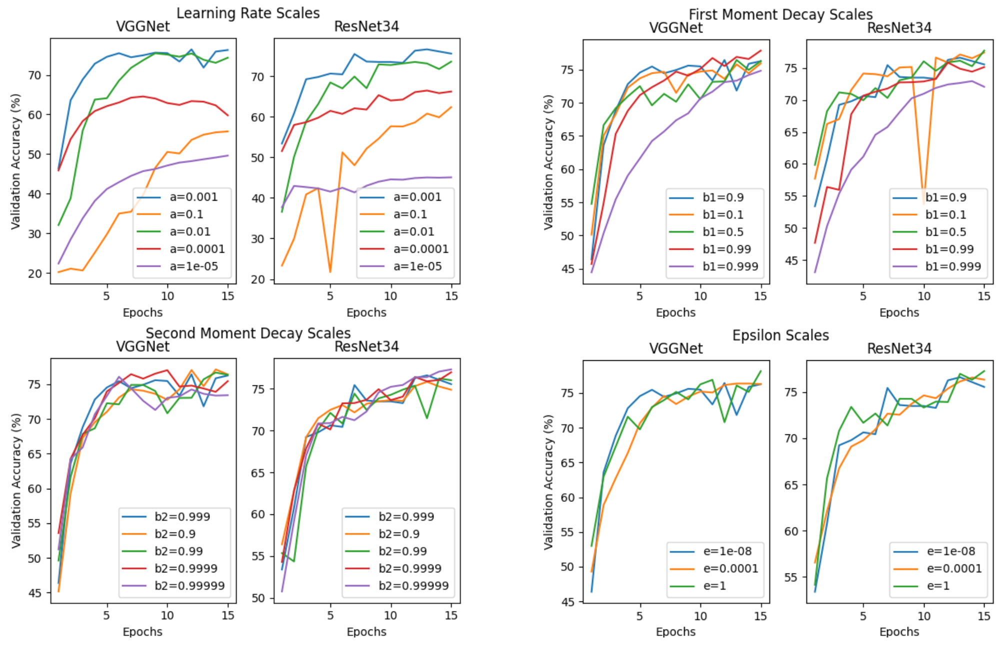

Figure 4: Validation accuracy curves for experiments on α (top left), β_1 (top right), β_2 (bottom left), and ϵ (bottom right).

it implies that the local minimum close to identity is not necessarily the best that can be achieved, though more experiments would be needed to investigate this further.

The experiments on the other three parameters all produce models with similar validation accuracy curves, though with some differences in validation loss curves (e.g., the default β_1 of 0.9 has the worst validation loss at the end but comparable validation accuracy with the other models).