Convolutional Neural Networks for CIFAR-10 multiclass image classification

Shivangi Patel
The University of Adelaide

Abstract

Convolutional Neural Networks have been popular for image classification analysis in the past decade. In this study, we compare different optimizers and investigate Cyclical Learning Rate scheduling method for classification of images in CIFAR-10 dataset. LR (Learning Rate) range test is employed to estimate the upper and lower bounds of learning rate scheduling. Our findings corroborate with those of Leslie et al [1] that cyclical learning rates do not greatly improve the performance when used in conjunction with adaptive gradient based optimizers. We also explore deeper architectures to increase classification accuracy. Image augmentation, Batch Normalization and Dropout methods are implemented to reduce overfitting observed with deeper neural networks. Finally, a CNN model with eight layers - 5 convolutional and 3 fully connected (FC) is trained with appropriate regularization, giving a classification accuracy of 83.2% on test data.

1. Introduction

Convolutional Neural networks (CNN) progressively extract higher level of representations of images. A typical CNN architecture consists of convolutional layers with intermediate pooling layers and concluded with one or more fully connected layers. While it is desirable to train neural networks with several layers to efficiently represent more complex mapping functions; the problem of vanishing or exploding gradients is a limiting factor. Replacement of *sigmoid* or *tanh* activation functions in hidden layers with *ReLU* – Rectified Linear Unit function [2] was a major algorithmic change that greatly improved performance of feed forward networks and made it possible to explore deeper networks. Deep convolutional neural networks with *ReLUs* train several times faster than their equivalents with *tanh* units [3]. Better procedures for weight initialization have also shown to be critical in prevention of layer activation outputs from exploding or vanishing through a forward pass in deep networks [4]. Another technique introduced in 2015 that seemed to dramatically impact optimization performance, was Batch Normalization [5]. The distribution of each layer's inputs changes during the course of training, a phenomenon termed as *Internal Covariate Shift*. Batch normalization stabilizes the network by rescaling and recentering the layer inputs; thereby mitigating the problem of *Internal Covariate Shift* and improving performance. Several researchers have also focused to address challenges in optimization of gradient descent, namely slow convergence or even divergence, and failure of escaping saddle points. Adaptive gradient based algorithms that adjust learning rates in response to gradients have been proposed [6], [7], [8]. A good adaptive algorithm will usually converge much faster than simple back-propagation with a poorly chosen fixed learning rate. Another alternative to choosing a fixed learning rate are adoption of learning rate schedules, with *time based* and step decay schedulers being widely popular. A method of Cyclical learning rates (CLR) proposed in 2017 [1], suggested that allowing the learning rate to rise and fall between reasonable bounds during training is beneficial overall, even though it may temporarily harm the network's performance. Finally, a plethora of ingenious deep neural network architectures were designed in the past decade - AlexNet [3], VGG [9], Inception [10], ResNet [11] to name a few that greatly enhanced the possibilities of deep learning.

2. Background

Learning rate is the most important hyperparameter to tune as it has a significant impact on the performance of a model – more so when using Stochastic Gradient Descent (SGD) for optimization. At a learning rate higher than the optimal value, the loss function will fail to converge, while a sub optimal value may result in an excessively long time to converge or even get stuck on a local minimum. Even with a standard learning rate decay scheme, tuning of an initial learning rate to use is difficult. Adaptive gradient based optimizers have therefore become very popular in recent years. These optimizers provide an element-wise scaling term on learning rates without the need to tune the learning rate manually. Adagrad [8] is the first proposed gradient based optimizer, which adapts the learning rate to the parameters. It performs smaller updates for parameters associated with frequently occurring features, and larger updates for parameters associated with infrequent features. It eliminates the need to manually tune the learning rate. Its main disadvantage is the accumulation of squared gradients in the denominator, causing the learning rate to shrink as the training progresses to a point that it becomes infinitesimally small to and is no longer able to optimize the loss function. Several variants Adadelta [7], Adam [6] , Adamax [6] etc., were proposed to fix this issue and have been successfully applied to various problems. However, a non uniform scaling and an aggressive adaptation of learning rate of adaptive gradient based algorithms have shown to result in poor generalization performance [12]. Learning Rate Schedulers have been proposed in recent years and have improved performances with state-of-the-art CNN architectures. Stochastic Gradient Descent with warm restarts is a learning rate scheduler [13],where the authors suggest to periodically simulate warm restarts of SGD, where in each restart the learning rate is initialized to some value and is scheduled to decrease in the form of a half cosine curve. Using Wide Residual neural networks (WRN), a test error of around 4% is reported with the SGD warm restart scheme on the CIFAR-10 dataset. Hyperbolic -tangent decay (HTD) method [14] is a variant of SGDR cosine curve based scheduling , reporting a test error of 5.6% (ResNet-110), 4.4% (Densenet-BC-250-24) and 4.2% (WRN-28-10) on CIFAR-10 dataset. Cyclical Learning Rate proposed by Leslie et al [1] is a method of varying learning rate in a rhythmic fashion during training, between a fixed pre-determined lower and upper bounds of learning rates. This eliminates the need to run a suite of experiments to determine the optimal learning rate. It is suggested that allowing the learning rate to rise and fall is beneficial overall, even though it might temporarily harm the network. As an intuition, with a low global or decaying learning rate, it is difficult to optimize the loss at saddle points; while with an increasing learning rate of CLR method, rapid traversal of saddle point plateaus is possible. Practically, it is likely that the optimal learning rate will be between the bounds and near optimal learning rates will be used throughout the training process. CIFAR-10 test errors of 6.7% and 5.1% using benchmark ResNet and DenseNet architectures respectively with CLR scheduling were reported. In this study, we investigate CLR based scheduling method with SGD and adaptive gradient based optimizers -Adagrad, Adam, Adadelta and Adamax using vanilla CNN architectures on CIFAR-10 dataset. With adaptive gradient based optimizers, we propose that fixing an upper and lower bound with CLR method may prevent them from aggressively adapting the learning rate; thereby improving their generalization ability.

3. Methodology

3.1. Dataset

Pytorch library is used in this study to configure CNNs for image classification in CIFAR-10 dataset, which is a labelled subset of 80 million tiny images dataset. It consists of 60000 32×32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. The classes are completely mutually exclusive. To monitor performance of models during training, a validation dataset consisting of 10000 images is generated by shuffling and splitting the train set. A utility function is defined to load and return multi-process iterators of train, validation and test sets over the CIFAR-10 dataset. An optional parameter to apply image augmentation on the train set invokes a data augmentation scheme, consisting of *Pytorch* built in methods *RandomCrop* and *RandomHorizontalFlip*. Unless otherwise specified, a batch size of 50 and a *CrossEntropyLoss* function is used throughout the study.

3.2. Cyclic Learning Rate

The CLR scheduling method varies learning rate in a rhythmic fashion during training, between a fixed predetermined lower and upper bounds of learning rates. In the original paper, the authors suggested three different policies of varying learning rates – a triangular window (linear), a Welch window (parabolic) and a Hann window (sinusoidal) all producing equivalent results. Therefore, the simplest function of linearly increasing and then decreasing i.e. a triangular window is used in this study (Figure 1. Blue lines

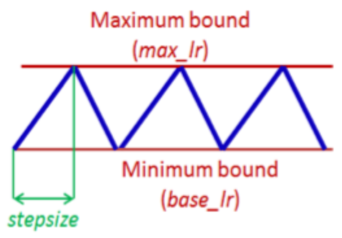

Figure 1. CLR - Triangular Policy (linear)

in Figure 1 represent learning rate values changing between lower and upper bounds. `Stepsize` is the number of iterations in half a cycle. The lower and upper boundary values are determined using the LR range test as suggested in [1]. Briefly, an LR range test is done by running the network for a few epochs over a `base_lr` and `max_lr`. For all the optimizers evaluated in this study, we exponentially range over a `base_lr` of 10^{-7} to a `max_lr` which is around 10 fold higher to the optimizer's default learning rate, for 2 epochs. After each iteration, the learning rate and losses are captured to plot an *lr-loss* graph. Figure 2 shows a typical lr-loss plot for an optimizer. The optimal learning rate is at the

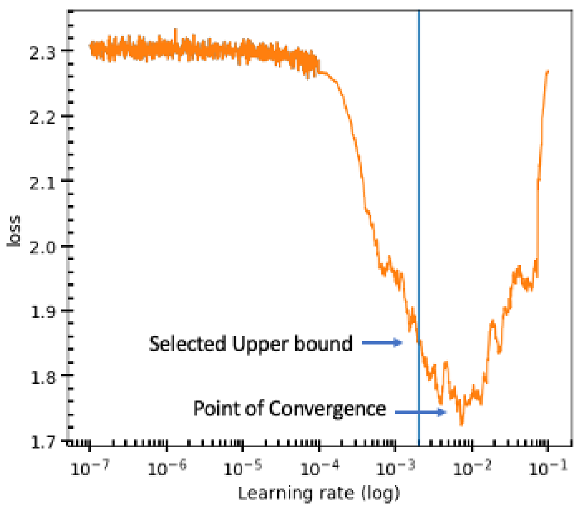

Figure 2. Typical lr-loss graph

point of convergence where loss is minimum. At learning rates beyond the optimal value, the model starts diverging. To leave a safe margin, we choose around 5-fold lower than the optimal learning rate as the upper bound. Lower bound is set as one-sixth of the upper bound. Once the lower and upper bounds of learning rate for each optimizer are determined, a cyclic learning rate schedule can be designed as shown in code snippet below,

```
local cycle = math.floor(1 +
epochCounter/(2 + stepsize))

local x = math.abs(epochCounter/stepsize
−2∗cycle+1)

local lr = opt.LR + (maxLR − opt.LR)
∗ math.max(0,(1−x))
```

where `opt.lr` is `base_lr`, `epochCounter` is number of epochs in training and `lr` is the computed learning rate. This code varies the learning rate between `base_lr` and `max_lr`. Learning rate during training is updated after each iteration by adding the scheduler below the optimizer step.

3.3. Optimizers and CNN architectures

We investigate the effect of applying CLR scheduling method with SGD and adaptive gradient based optimizers - Adagrad, Adam, Adadelta and Adamax using simple CNN architecture (CNN1, see Table 1) and training models for 25 epochs. The optimizer with best performance in terms of accuracy and generalisation ability is used in later experiments. We then evaluate various depths of CNN architectures, alongwith implementation of regularisation techniques - image augmentation, Batch normalization and Dropout method. The CNN architecture with best performance in terms of accuracy and generalisation ability is then used to train a model for 100 epochs and evaluated on test data. Table 1 summarizes the key differences in the four network architectures designed in this study.

Network	Layers	Regularization	Parameters
CNN1	5 - 2C,3FC	None	62,006
CNN2	6 - 3C,3FC	None	155,070
CNN3	6 - 3C,3FC	BN	155,686
CNN4	8 - 5C,3FC	BN,D	3,018,906

Table 1. Summary of CNN architectures, C - Convolution, FC - Fully Connected, BN - Batch Normalization, D - Dropout

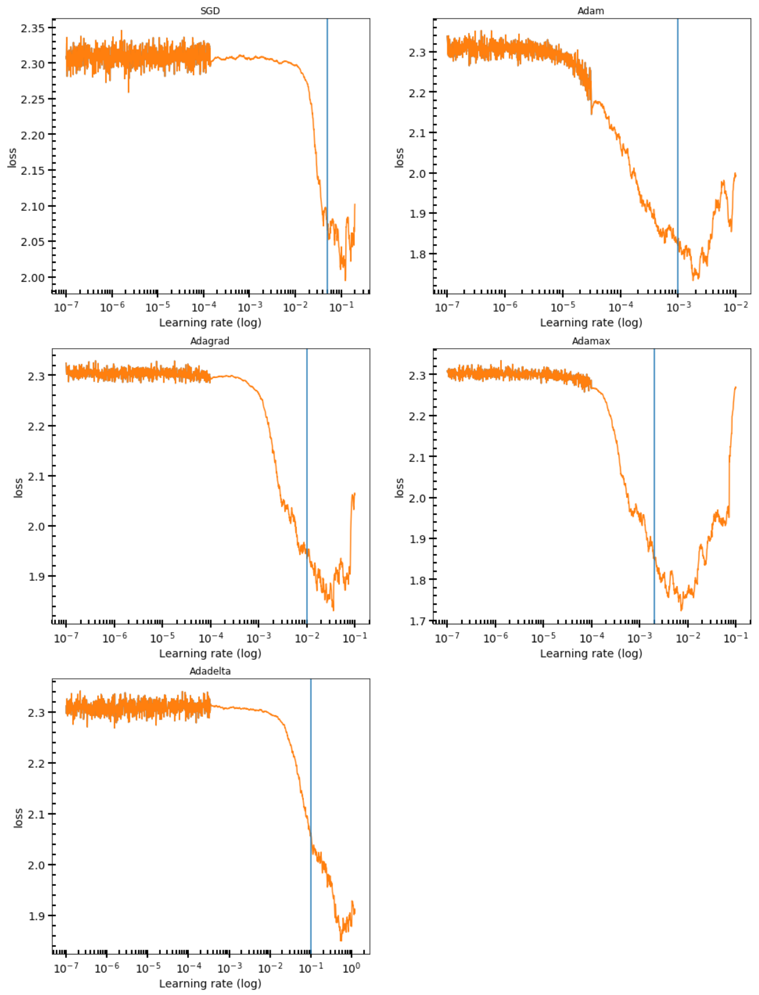

Figure 3. LR range test for optimizers, blue lines represent chosen upper bound.

4. Experimentation

We opt to employ CLR scheduling method over a global or monotonically decaying learning rate for each of the optimizers - SGD, Adam, Adagrad, Adadelta and Adamax.

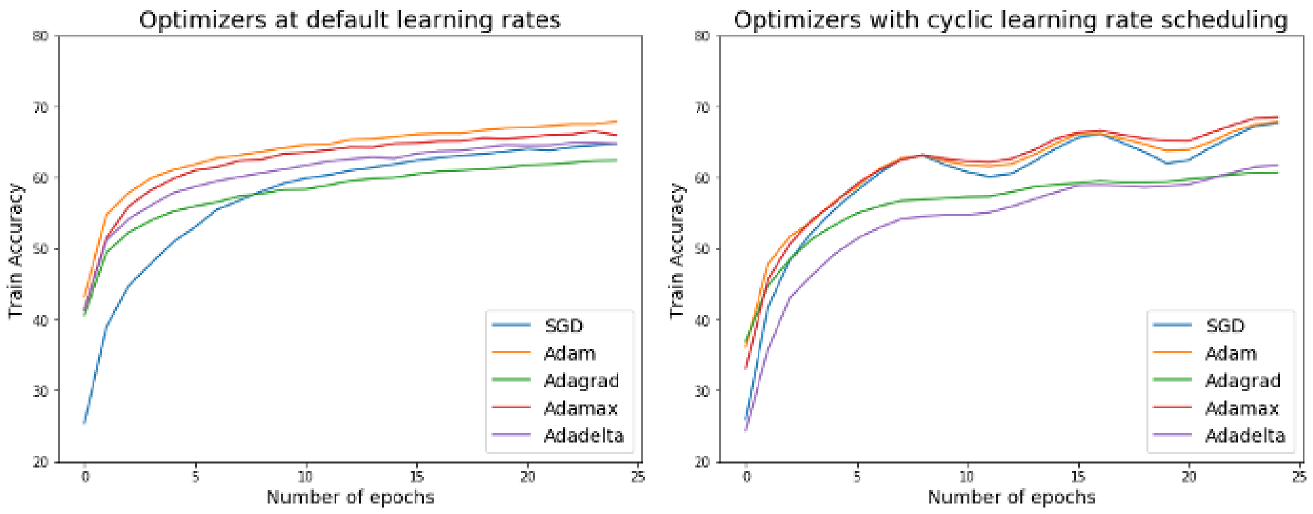

Figure 4. Comparison of optimizers at CLR schedule and global default learning rate

Figure 3 shows results of LR range test conducted for each optimizer by running CNN1 (5 layers - 2C, 3FC) network over 2 epochs. Based on the upper and lower bound determined for each optimizer using LR range test, a CLR schedule is designed for all optimizers. Figure 4 shows accuracies obtained by training for 25 epochs with optimizers on CLR schedule or using a global default learning rate. Except Adadelta, the overall training accuracies with all optimizers employing a CLR or global default learning rate are equivalent. Only with SGD, a slightly faster convergence is observed when using CLR method. Our findings corroborate to Leslie et al [1], who also observed that CLR method does not significantly alter performance or convergence time when used in conjunction with adaptive gradient based optimizers. Adamax, Adam and SGD optimizers give highest training accuracy with CNN1. The generalization gap with Adamax optimizer is slightly lower when compared to the other two and therefore chosen as the best optimizer (data not shown). However, the model has an overall low training accuracy indicating that it is underfitting the data. Increasing the depth of the network with additional number of filters or nodes in dense layers would enable the model to map complex functions in the dataset. To further increase the complexity of the model, CNN2 network is designed. CNN2 (6 layers - 3C, 3FC) network has an additional convolutional layer with 64 filters. To match dimensions of the connected dense layer, the number of nodes are increased accordingly. With the added complexity in the neural network architecture, an improvement in training accuracy (around 11% over CNN1) can be observed as the model is trained for 25 epochs, however the generalization gap increases after 10 epochs as seen in Figure 6a. We therefore investigate various regularization techniques with CNN2 network to improve its generalization ability. Image augmentation is a well-known technique used to address overfitting. It artificially expands the training data by performing affine transformations, allowing the model to learn features that are invariant to their location in the original image.Figure 6 shows a considerable improvement in generalisation with image augmentation, albeit with a lower training accuracy. Batch normalization, although mainly used as a regularization technique, is also known to achieve faster training times. We therefore design CNN3, which is the same depth as CNN2 but with Batch Normalization implemented before every non linear activation in the network. We observe a slightly faster convergence rate and increase in training accuracy with implementation of Batch Normalization (Figure 5).

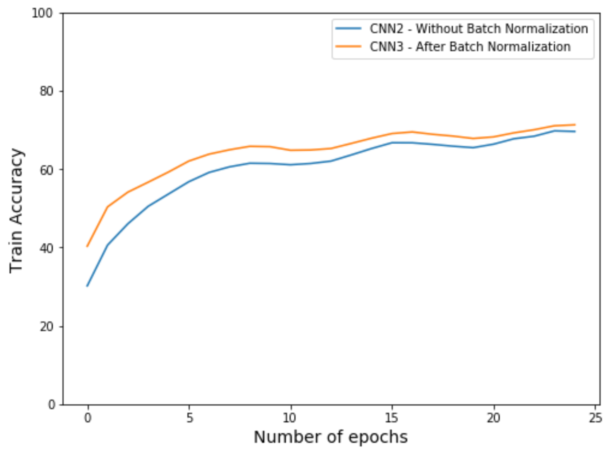

Figure 5. Effect of Batch Normalization on training

To further improve training accuracy we investigate a deeper network CNN4 (8 layers – 5C, 3FC). With increased number of filters in convolutional layers and nodes in dense

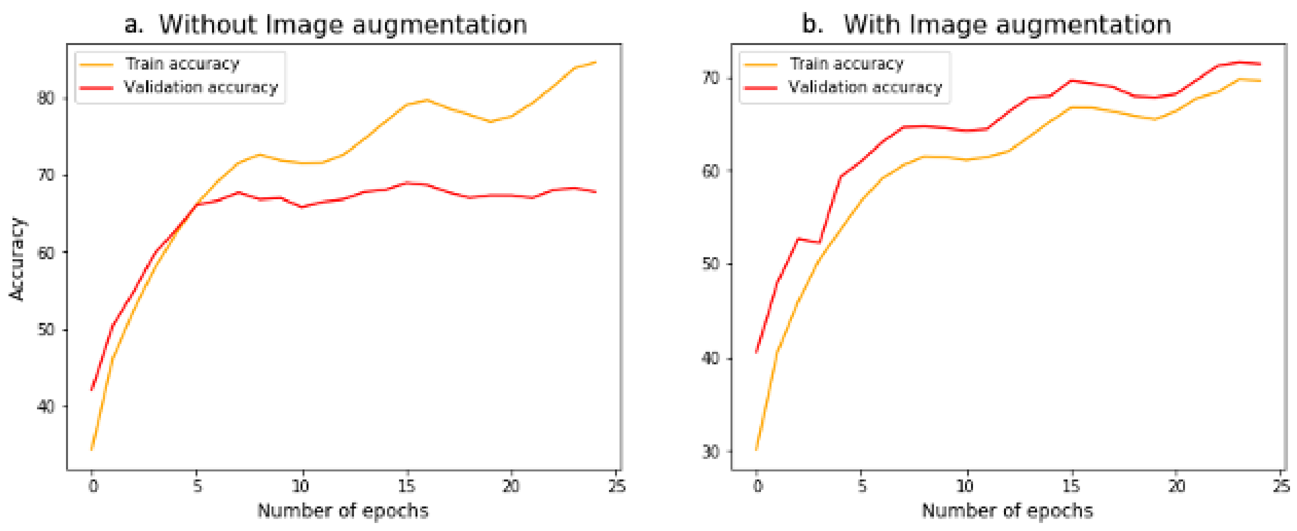

Figure 6. Effect of image augmentation in generalization ability

layers, there is a further increase in the complexity of the network and it is likely to overfit. Therefore, in addition to image augmentation and Batch Normalization, we also implement Dropout technique of regularization on the first two fully connected layers. When trained for 25 epochs using CNN4, the model shows a significant increase in accuracy and very good generalization ability (data not shown). A comparison of accuracies obtained after image augmentation and training for 25 epochs using the four network architectures is shown in Figure 7. We observe a significant increase in model performance with the depth of neural network architecture.

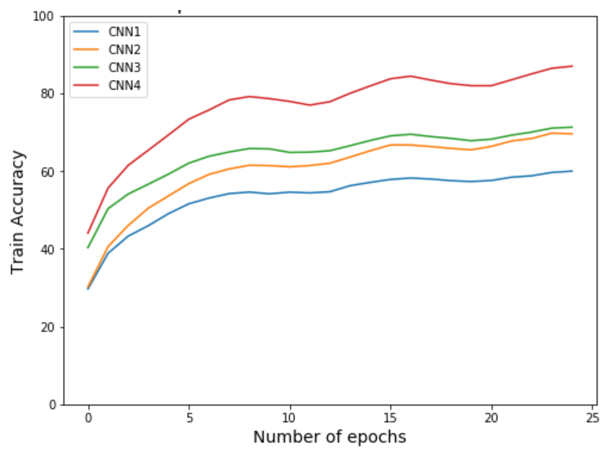

Figure 7. Comparison of network architectures

Finally, Figure 8 shows accuracy curves of model trained for 100 epochs using the CNN4 architecture with Adamax optimizer on a CLR scheduling method, and image augmentation of training samples. The generalization gap slightly widens after 30 epochs. Upon evaluation on test set, the trained model is able to classify images with 83.2% accuracy.

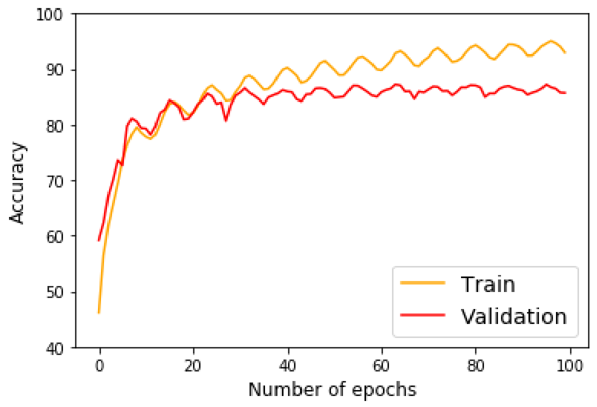

Figure 8. Accuracy curves for model with CNN4 architecture

5. Conclusion

We investigated a Cyclical Learning Rate schedule for SGD, Adagrad, Adam, Adadelta and Adamax on CIFAR-10 dataset. Applying CLR gives a slightly faster convergence with SGD relative to using its default learning rate, however the performance is mostly unaltered for adaptive gradient based optimizers (except Adadelta, where performance deteriorates with CLR). We compared different CNN architectures with progressively increasing depth and numbers of filters and nodes (for dense layers) for their accuracy and generalization ability. As the depth of a network increases, the model is able to map complex functions with significant increase in accuracy but with a high generalisation error. Applying regularisation techniques like image augmentation, Batch Normalization and Dropout helped improve generalisation ability of the deeper networks. Our best model is able to classify images in CIFAR-10 dataset with 83.2% accuracy.

6. Github

For code, please refer to Github.

References

[1] L. N. Smith, "Cyclical learning rates for training neural networks," in *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2017, pp. 464–472.

[2] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ser. ICML'10. Madison, WI, USA: Omnipress, 2010, p. 807–814.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS'12. Red Hook, NY, USA: Curran Associates Inc., 2012, p. 1097–1105.

[4] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," ser. Proceedings of Machine Learning Research, Y. W. Teh and M. Titterington, Eds., vol. 9. Chia Laguna Resort, Sardinia, Italy: JMLR Workshop and Conference Proceedings, 13–15 May 2010, pp. 249–256. [Online]. Available: http://proceedings.mlr.press/v9/glorot10a.html

[5] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, ser. JMLR Workshop and Conference Proceedings, F. R. Bach and D. M. Blei, Eds., vol. 37. JMLR.org, 2015, pp. 448–456. [Online]. Available: http://proceedings.mlr.press/v37/ioffe15.html

[6] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014.

[7] M. D. Zeiler, "Adadelta: An adaptive learning rate method," 2012.

[8] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. 61, pp. 2121–2159, 2011. [Online]. Available: http://jmlr.org/papers/v12/duchi11a.html

[9] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014.

[10] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *CoRR*, vol. abs/1409.4842, 2014. [Online]. Available: http://arxiv.org/abs/1409.4842

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[12] N. S. Keskar and R. Socher, "Improving generalization performance by switching from adam to SGD," *CoRR*, vol. abs/1712.07628, 2017. [Online]. Available: http://arxiv.org/abs/1712.07628

[13] I. Loshchilov and F. Hutter, "Sgdr: Stochastic gradient descent with warm restarts," in *ICLR*, 2017.

[14] B. Y. Hsueh, W. Li, and I. Wu, "Stochastic gradient descent with hyperbolic-tangent decay," *CoRR*, vol. abs/1806.01593, 2018. [Online]. Available: http://arxiv.org/abs/1806.01593