Residual Network Design

ArunKumar Nachimuthu Palanichamy, Shoban Venugopal Palani,
Viswanathan Babu Chidambaram Ayyappan
Department of Electrical and Computer Engineering, New York University
{an3333, sv2244, vc2173}@nyu.edu

Abstract

A ResNet architecture [1] is designed for classification of the CIFAR-10 dataset [7] with an objective of maximizing the test accuracy while limiting the total number of trainable parameters within 5 million. The test loss produced by various optimization techniques such as Lookahead optimizer with Adam [2] and SGD, Adam [3], Adagrad [4], Adadelta [5] etc., are compared to obtain the optimal algorithm suitable for the given problem. Data augmentation methods are implemented to improve performance on the test dataset. Finally, it is demonstrated that the Lookahead with adam algorithm outperforms the other considered algorithms in terms of test loss.

1 Introduction

The existence of Residual networks [1] came into place when deeper neural networks became harder to train and accuracy started saturating. Since, it is not a problem of overfitting, using dropouts will not work. Residual networks are still deeper and the idea behind it is to connect the output of the previous layers to the output of new layers by adding skip connections. This technique addresses the problem of vanishing gradients in deep layers and overcomes poor accuracy.

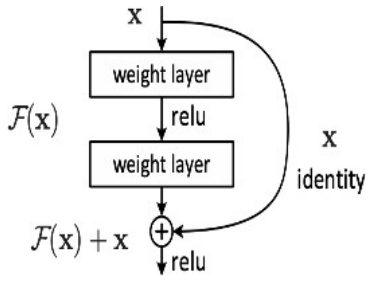

Figure 1: Residual Skip Connection in a Neural Network [1]

In this paper, to improve the test accuracy, a ResNet architecture is employed on the CIFAR-10 dataset [7]. The CIFAR-10 dataset contains 50,000 training images and 10,000 test images. The dataset contains 10 different classes and each class has 6000 images in them. Due to the low resolution of the images in this dataset, various algorithms are tried out with comparatively less computational time and hence, it is one of most widely used dataset for testing novel algorithms and network architectures.

A Residual Network with custom specifications such as Residual block size, kernel size, stride and

other hyperparameters is designed and the Lookahead technique is used to improve the performance of Adam [2] and finally, the test performance of different optimization techniques used on the network are compared to find the optimal one.

2 Methodology

This section explain the step by step process of the network, any techniques used and the hyperparameters chosen for the model.

2.1 Dataset

The CIFAR-10 dataset contains 10 different classes such as cars, dogs, trucks, ships etc. The dataset contains 50,000 training images and 10,000 test images each a 32x32 color image.

2.2 Data Augmentation

Data Augmentation [12, 14] is the process of transforming the original image using different techniques. Augmenting the data by doing these transforms will improve the accuracy of the model. Since the training data might not have all the features in an image, thus by doing these transformations, the model will learn more features. Hence, whenthe model sees it on test data, it performs well. For this model, the following transforms are performed [9]:

1. **Random Crop:** A 32x32 random crop of the image is done with a padding of 4, since all the images in the dataset are 32x32. This step is to make sure that the size of the all the images that are sent to the model are maintained constant. This is to add white space to the edges of the images.
2. **Random Horizontal Flip:** This technique horizontally flips the image which will help the model learn more features from the image, since an image is fed to the network along with its mirror image.
3. **To Tensor:** Converting the images to tensors of shape (C, H, W) for PyTorch to process.
4. **Normalize:** Normalization of the images are performed by subtracting each image from the mean of the image set and dividing by the standard deviation. Since, CIFAR-10 contains colored images,this is done for all 3 channels. Thus it convert images or arrays in the range [0, 255] to [0, 1].

2.3 Data Loaders

The data loaders in PyTorch helps in carrying out all the data transformations and augmentation methods in real time and feed it to the network even for a large dataset. The batch sizechosen is as 32 and hence, PyTorch takes 32 images in a batch and does all the transformations and feeds it to the network. The data loaders have several inbuilt arguments that aids in data handling such as the number of workers which indicated the number of sub-processes to use for data loading. To take full advantage of the GPU, but not exploit it fully, number of workers is selected as 2. The drop_last argument helps drop the last incomplete batch i.e., when the dataset is indivisible by the batch size. This is set True only for Training and not for validation. Setting the shuffle argument as Truefor training data helps the model to converge. Shuffling Validation set is not necessary, since it does not help the model.

2.4 Parameters of the Network

The parameters of the network are selected considering the constraint of a maximum of 5 million trainable parameters. By observing the train and test loss curves and by evaluating the train and test accuracies as well, the fit of the network is inferred and the hyperparameters of the network are tuned accordingly. The number of layers, number of ResNet blocks, batch size etc., are chosen on a trial and error basis with optimum test accuracy as the primary focus. Early stopping technique is utilized to prevent overfitting.

2.5 Optimizers

For this model, a special technique for Adam adaptive optimization method known as Lookahead is utilized. Lookahead is a type of stochastic optimizer that iteratively updates two sets of weights: "fast" and "slow". Intuitively, the algorithm chooses a search direction by looking ahead at the sequence of fast weights generated by another optimizer. Lookahead first updates the "fast weights" k times using any standard optimizer in its inner loop before updating the "slow weights" once in the direction of the final fast weights. The Lookahead method is less sensitive to sub-optimal hyperparameters and therefore lessens the need for extensive hyperparameter tuning. By using Lookahead with inner optimizers such as Adam, faster convergence can be achieved across different deep learning tasks with minimal computational requirements.

Algorithm 1 Lookahead Optimizer:

Require: Initial parameters ϕ_0, objective function L
Require: Synchronization period k, slow weights step size α, optimizer A
  for $t = 1, 2, \ldots$ **do**
    Synchronize parameters $\theta_{t,0} \leftarrow \phi_{t-1}$
    for $i = 1, 2, \ldots, k$ **do**
      sample minibatch of data $d \sim \mathcal{D}$
      $\theta_{t,i} \leftarrow \theta_{t,i-1} + A(L, \theta_{t,i-1}, d)$
    end for
    Perform outer update $\phi_t \leftarrow \phi_{t-1} + \alpha(\theta_{t,k} - \phi_{t-1})$
  end for
  return parameters ϕ

Figure 2: Algorithm for Lookahead Optimizer [2]

Lookahead algorithm computes weight updates by looking ahead at the sequence of "fast weights" generated by Adam or SGD. In this model, after experimentation with multiple optimizers, the best accuracy was obtained by using Lookahead with Adam.

2.6 Training

Now, the model and other hyperparameters are set, next step is to train the residual neural network. Data from the CIFAR-10 dataset is fed to the network through the Data Loaders with augmentation, and other chosen hyperparameters. The loss function chosen is Cross Entropy since the decision boundary in classification task is large. From a probabilistic point of view, the cross-entropy arises as the natural cost function to maximize the likelihood of classifying the input data correctly. Then, Adam optimizer is initialized and given as an input to the Lookahead class. Also, in this optimization technique, the learning rate automatically adapts as the optimizer tunes the learning rate during training. Optimum Number of epochs is selected by doing a callback technique [14] in PyTorch. The utilized callback technique includes an early stopping procedure [15, 10] that is triggered when the test loss increases in consecutive iterations. This saves a lot of time and computation, and it also prevents the model from overfitting since it checks for signs of overfitting after each iteration. L2 Regularization technique [13] is employed as well to improve the test performance.

3 Results

By evaluating the CIFAR-10 dataset [7] with a Residual Network employing different algorithms like lookahead optimizer with Adam [2] and SGD, Adam [3], Adagrad [4], Adadelta [5] etc., the table 1 has been recorded, indicating the test loss of each algorithm. The test losses presented in table 1 are obtained after running the network with the following hyperparameters: batch size – 32, ResNet block size – (2, 2, 2, 2), learning rate – 0.01, channels – (32, 64, 128, 256) and 25 epochs. The total trainable parameters of this network are 2,797,610, calculated using [11]. It is observed that the lookahead optimizer with Adam gives the best performance on the test data.

Table 1: Test accuracies of optimizers for 25 epochs

OPTIMIZER	TEST LOSS	RUN TIME
Lookahead with Adam	0.3974	15m 13s
Adagrad	0.4473	14m 04s
Adam	0.4167	15m 09s
Adadelta	0.6145	15m 48s
Lookahead with SGD	0.4549	13m 44s
NAdam	0.4047	15m 36s

3.1 Lookahead with adam optimizer

The train and test loss curve versus the number of epochs of the ResNet running on the lookahead with Adam optimizer [2, 8] without regularization is given in figure 3. It can be seen in the figure that the network starts to overfit the data.

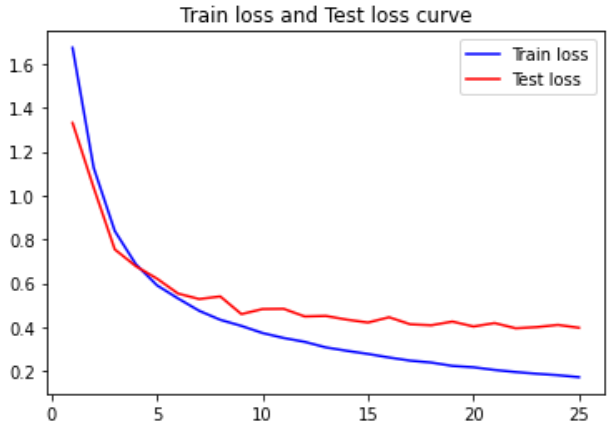

Figure 3: Train and test loss of lookahead without regularization

3.2 Adam optimizer

In figure 4, the loss curves versus the number of epochs of the ResNet running on Adam optimizer [3] is given where the test loss curve reaches a value of 0.4167 at the 25th epoch.

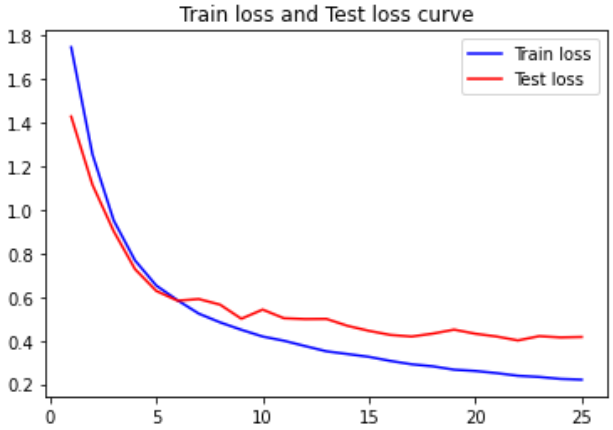

Figure 4: Train and test loss of Adam

3.3 Adagrad optimizer

The figure 5 represents the train and test loss curves of the Adagrad optimizer [4] and a test loss of 0.4473 is obtained at the 25th epoch.

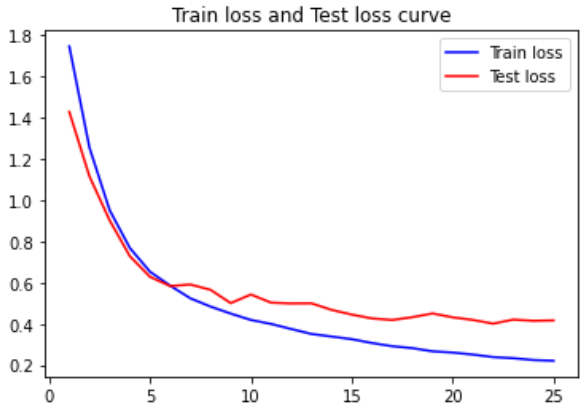

Figure 5: Train and test loss of Adam

3.4 Adagrad optimizer

The figure 6 represents the train and test loss curves of the Adagrad optimizer [4] and a test loss of 0.4473 is obtained at the 25th epoch.

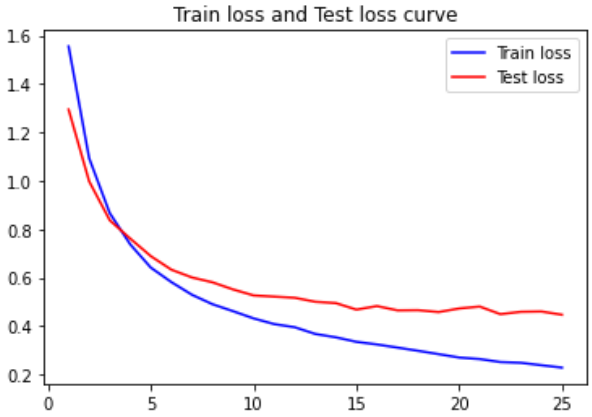

Figure 6: Train and test loss of Adagrad

3.5 Lookahead with SGD optimizer

The loss curves of the lookahead with SGD optimizer is given in Figure 7 where the test data loss is 0.4549 at the 25th epoch.

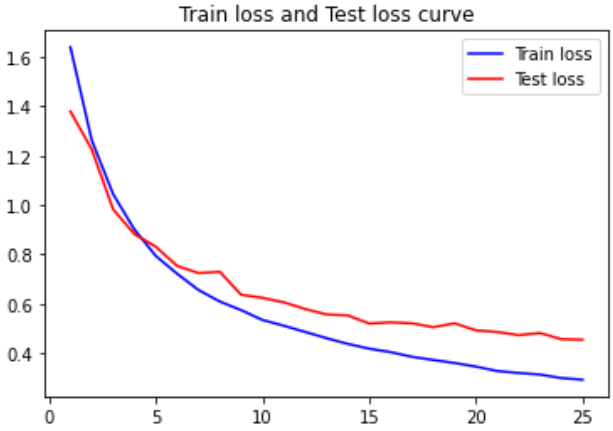

Figure 7: Train and test loss of Lookahead with SGD optimizer

3.6 Adadelta

Figure 8 illustrates the train and test loss curves of the Adadelta algorithm [5] and the test loss is 0.6145 at the 25th epoch.

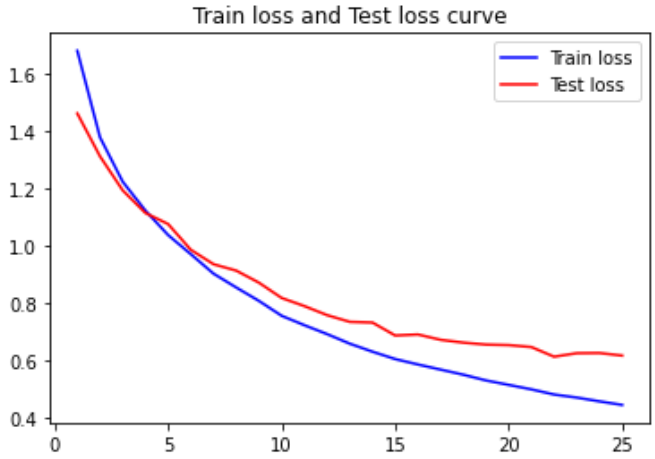

Figure 8: Train and test loss of Adadelta optimizer

3.7 NAdam

The Nesterov Momentum with Adam algorithm's [6] performance on the train and test data is given in Figure 9 and the test loss is 0.4047 at the 25th epoch.

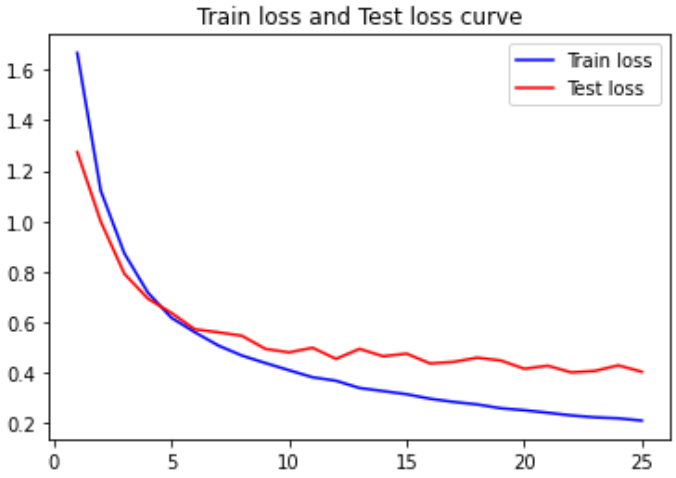

Figure 9: Train and test loss of NAdam

3.8 Final Network

After inferring that the lookahead algorithm along with Adam optimizer gives the best performance, L2 regularisation with a weight decay of 1e-5 is added, the hyperparameters of the network are further tuned and this algorithm is utilized on the network to produce a better test accuracy. The total number of trainable parameters are 4.9M. The final network has the following hyperparameters: batch size – 32, ResNet block size – (2, 1, 1, 1), learning rate – 0.001, channels – (64, 128, 256, 512) and 15 epochs which gives a test accuracy of 87.5% and a test loss of 0.3777 at the end of the 15th epoch. The train and test loss curve of the final network is given in Figure 10.

Table 2: Train and Test Loss of the Final Network for 15 epochs

EPOCH	TRAIN LOSS	TEST LOSS
1	1.3073111037591356	1.010189284436619
2	0.8506798194709416	0.8252702301112227
3	0.67352721802938	0.6303260532050087
4	0.5891190457531035	0.5817013472413864
5	0.5165436850209624	0.5560842960977707
6	0.4702143781984203	0.5040384490078631
7	0.42081794992718896	0.49089266209842297
8	0.38811594219675594	0.47125625119993864
9	0.366201685357567	0.4336371939546003
10	0.3354785827163812	0.44323939887193825
11	0.31416281437436916	0.4239874416218398
12	0.29674022302995035	0.41948513904461465
13	0.2752072756821421	0.42380973606254346
14	0.25914280144938967	0.3814625668639954
15	0.24644231023779675	0.3776631119152227

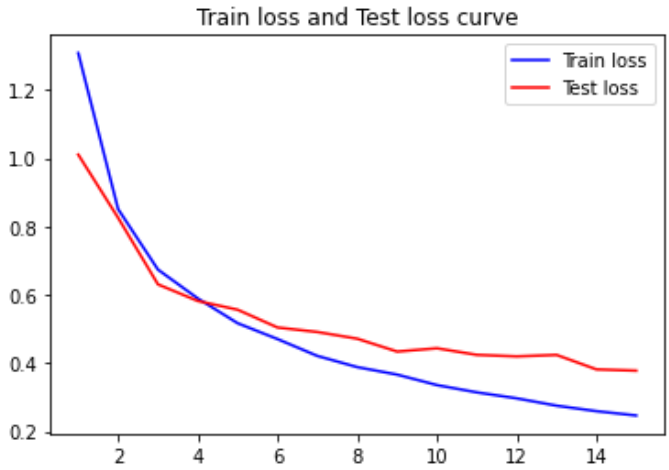

Figure 10: Train and test loss of the final network

4 Conclusion

After careful analysis of different optimization algorithms, hyperparameter tuning and evaluation of data handling and training techniques like data augmentation and early stopping callback, it is concluded that the lookahead method in combination with the Adam algorithm [2] employed on a residual network with trainable parameters less than 5 million, provides better test performance in comparison with the other considered algorithms on the CIFAR-10 image dataset.

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[2] Michael R. Zhang, James Lucas, Jimmy Ba, and Geoffrey E. Hinton. Lookahead optimizer: k steps forward, 1 step back. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada, pp. 9593–9604, 2019b.

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

[4] John Duchi, Elad Hazan, and Yoram Singer. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. Journal of Machine Learning Research, 12:2121–2159, 2011.

[5] Zeiler, M. D. (2012). ADADELTA: An adaptive learning rate method. arXiv:1212.5701 [cs.LG].

[6] Timothy Dozat. Incorporating Nesterov Momentum into Adam. ICLR Workshop, (1):2013–2016, 2016.

[7] Krizhevsky, Alex, Vinod Nair, and Geoffrey Hinton. "The CIFAR-10 dataset." *online: http://www.cs.toronto.edu/kriz/cifar.html* 55, no. 5 (2014).

[8]https://github.com/lonePatient/lookahead_PyTorch/blob/1055128057408fe8533ffa30654551a317f07f0a/optimizer.py#L7

[9] https://androidkt.com/PyTorch-image-augmentation-using-transforms/

[10] https://www.kaggle.com/general/178486

[11] https://gist.github.com/thongonary/026210fc186eb5056f2b6f1ca362d912

[12] L. Perez and J. Wang. The effectiveness of data augmentation in image classification using deep learning. arXiv preprint arXiv:1712.04621, 2017.

[13] Ng A Y 2004 Feature selection, l1 versus l2 regularization, and rotational invariance *Proc. 21st Int. Conf. on Machine Learning* (New York, NY: ACM) p 78

[14] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. Journal of Big Data, 6(1):60, 2019.

[15] Rich Caruana Steve Lawrence Lee Giles. 2001. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In Advances in Neural Information Processing Systems 13: Proceedings of the 2000 Conference, volume 13, page 402. MIT Press

Github Repository Link:

The following github link includes all the codes , report and a readme.md file explaining how to reproduce and run the code.