Mini-Project 1: Residual Network Design

Kunal Kashyap
kk4564
N19197590

Mukta Maheshwari
mm11070
N18434644

Neelanchal Gahalot
ng2436
N19484651

Abstract

Residual Network (ResNet) architecture is commonly used for image classification and is an example of CNN used extensively when it comes to scaling thousands of layers. Each layer adds to better performance but lends itself to diminishing returns, the deeper we traverse through the architecture. We try to reproduce better results with Wide Residual Networks while trying to cut down on computation. The scope of this project is restricted to CIFAR-10 dataset and ResNet. It becomes imperative to curtail the total number of hyper-parameters for applications constrained with limited data storage – IOT devices serve as a good example. The project institutes with deep ResNet architecture, and with each iterative experimentation, we augment our dataset with a better (often wider architecture) to improve our test accuracy on CIFAR-10. Our code is available at https://github.com/kunalkashyap855/wide-resnet-cifar10.

1 Introduction

Residual Network makes it possible to train up to hundreds or even thousands of layers and still achieves compelling performance and as a CNN, it has seen incredible proliferation over the years. Its powerful representational ability has produced seminal results in Computer Vision applications. Residual Neural Network (ResNet) is an Artificial Neural Network, which makes use of skip connections to jump over layers. A ResNet may contain N residual layers, and each residual layer may contain more than one residual block. A residual block contains two convolutional layers with skip connection from input to output. Formal definitions have been mentioned in the problem statement itself. All experimentation is done CIFAR-10 dataset, which is a collection of 60,000 32x32 color images in 10 different classes. The 10 different classes represent airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. There are 6,000 images of each class. The goal is to maximize test accuracy on CIFAR 10 dataset, used as a benchmark for Computer Vision applications.

Choice of hyper-parameters for Neural Networks is crucial for obtaining best training results and this report summarizes all experimentations done to tune these parameters for highest accuracy. The cap for trainable parameters is set at 5 million with flexibility in choosing different data augmentation strategies. The model can be trained with any optimizer and regularization scheme of choice.

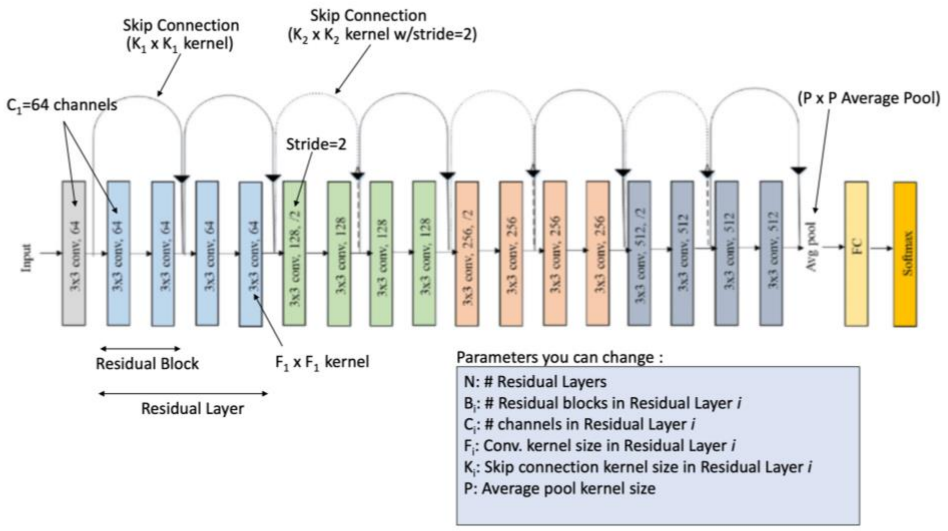
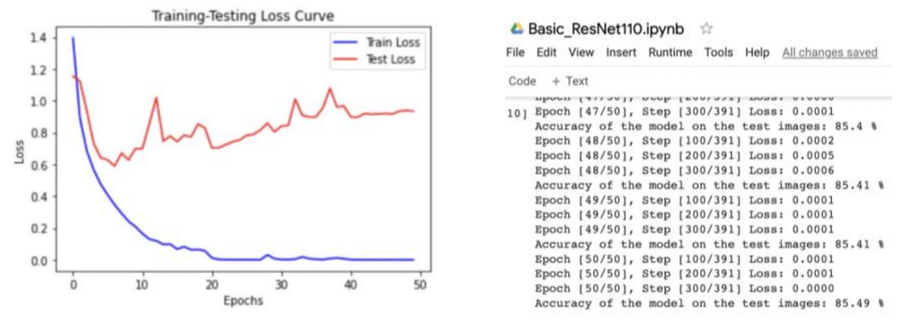


Figure 1: Sample Figure Caption

2 Methodology

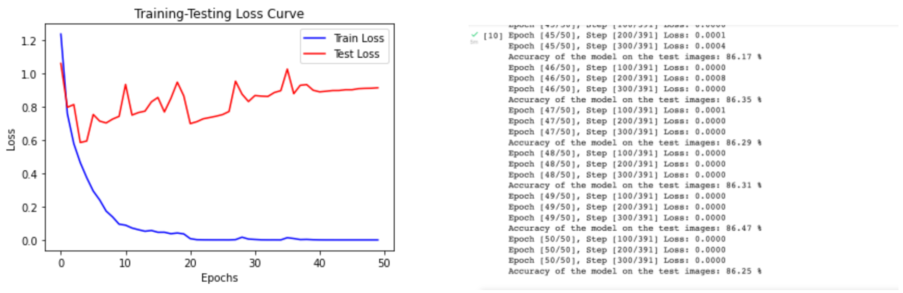
ResNet can employ a very large number of layers to train over a dataset and various techniques (like well-defined initialization strategies) have been proposed for deeper networks [1]. We set out to train our first model using 110 total layers (3 residual layers with 18 residual blocks each, each residual block has 2 convolutional layers). Batch size for all our models is 128. But as demonstrated in the work of Zagoruyko et al. [2], a deeper architecture may not the best implementation and can run problems like vanishing gradient descent [10].



```
10] Epoch [47/50], Step [300/391] Loss: 0.0001
    Accuracy of the model on the test images: 85.4 %
    Epoch [48/50], Step [100/391] Loss: 0.0002
    Epoch [48/50], Step [200/391] Loss: 0.0005
    Epoch [48/50], Step [300/391] Loss: 0.0006
    Accuracy of the model on the test images: 85.41 %
    Epoch [49/50], Step [100/391] Loss: 0.0001
    Epoch [49/50], Step [200/391] Loss: 0.0001
    Epoch [49/50], Step [300/391] Loss: 0.0001
    Accuracy of the model on the test images: 85.41 %
    Epoch [50/50], Step [100/391] Loss: 0.0001
    Epoch [50/50], Step [200/391] Loss: 0.0001
    Epoch [50/50], Step [300/391] Loss: 0.0000
    Accuracy of the model on the test images: 85.49 %
```

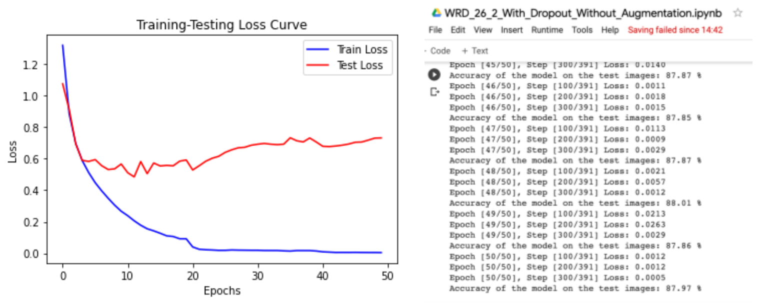
Figure 2: ResNet-110 (# Params: 1.73M)

Also, a deeper architecture takes much longer to run and makes computation more expensive. Cutting down on the number of layers for our next model, we use a total of 26 layers (4 residual layers, with 3 residual blocks in each, each block with 2 convolutional layers). A wider network presents a much better accuracy with runs faster. To further improve upon our wide network, we use a dropout layer, which can demonstrably yield consistent, even state of the art results.



```
[10] Epoch [45/50], Step [200/391] Loss: 0.0001
     Epoch [45/50], Step [300/391] Loss: 0.0004
     Accuracy of the model on the test images: 86.17 %
     Epoch [46/50], Step [100/391] Loss: 0.0000
     Epoch [46/50], Step [200/391] Loss: 0.0008
     Epoch [46/50], Step [300/391] Loss: 0.0000
     Accuracy of the model on the test images: 86.35 %
     Epoch [47/50], Step [100/391] Loss: 0.0001
     Epoch [47/50], Step [200/391] Loss: 0.0000
     Epoch [47/50], Step [300/391] Loss: 0.0000
     Accuracy of the model on the test images: 86.29 %
     Epoch [48/50], Step [100/391] Loss: 0.0000
     Epoch [48/50], Step [200/391] Loss: 0.0000
     Epoch [48/50], Step [300/391] Loss: 0.0000
     Accuracy of the model on the test images: 86.31 %
     Epoch [49/50], Step [100/391] Loss: 0.0000
     Epoch [49/50], Step [200/391] Loss: 0.0000
     Epoch [49/50], Step [300/391] Loss: 0.0000
     Accuracy of the model on the test images: 86.47 %
     Epoch [50/50], Step [100/391] Loss: 0.0000
     Epoch [50/50], Step [200/391] Loss: 0.0000
     Epoch [50/50], Step [300/391] Loss: 0.0000
     Accuracy of the model on the test images: 86.25 %
```

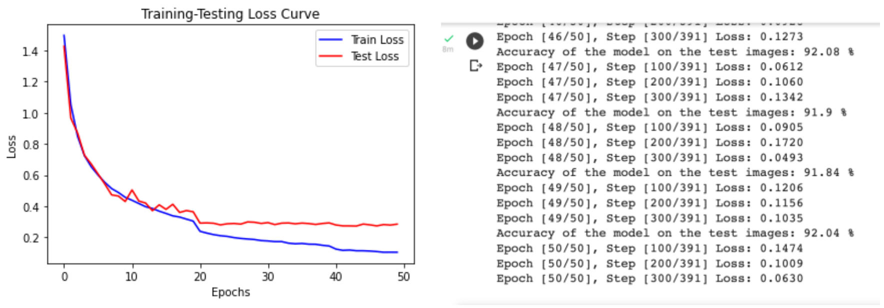
Figure 3: WRN-26-2 (# Params: 4.32M)



```
Accuracy of the model on the test images: 87.87 %
Epoch [46/50], Step [100/391] Loss: 0.0011
Epoch [46/50], Step [200/391] Loss: 0.0018
Epoch [46/50], Step [300/391] Loss: 0.0015
Accuracy of the model on the test images: 87.85 %
Epoch [47/50], Step [100/391] Loss: 0.0113
Epoch [47/50], Step [200/391] Loss: 0.0009
Epoch [47/50], Step [300/391] Loss: 0.0029
Accuracy of the model on the test images: 87.87 %
Epoch [48/50], Step [100/391] Loss: 0.0021
Epoch [48/50], Step [200/391] Loss: 0.0057
Epoch [48/50], Step [300/391] Loss: 0.0012
Accuracy of the model on the test images: 88.01 %
Epoch [49/50], Step [100/391] Loss: 0.0213
Epoch [49/50], Step [200/391] Loss: 0.0263
Epoch [49/50], Step [300/391] Loss: 0.0029
Accuracy of the model on the test images: 87.86 %
Epoch [50/50], Step [100/391] Loss: 0.0012
Epoch [50/50], Step [200/391] Loss: 0.0012
Epoch [50/50], Step [300/391] Loss: 0.0005
Accuracy of the model on the test images: 87.97 %
```

Figure 4: WRN-26-2 with Dropout (# Params: 4.32M)

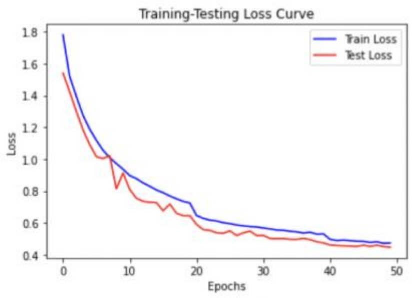
To further improve our results, we made use of two widely used data augmentation (techniques used to increase the amount of data by adding slightly modified copies of already existing data or newly created synthetic data from existing data techniques), namely random horizontal flip (horizontal flip is better than vertical flip) [3] and random crop. We do it from image padded by 4 pixels on each side and we use reflections of original image to fill missing pixels. With these two changes (dropout layer and data augmentation), we see significant improvement in our accuracy, with test accuracy reaching 92%.

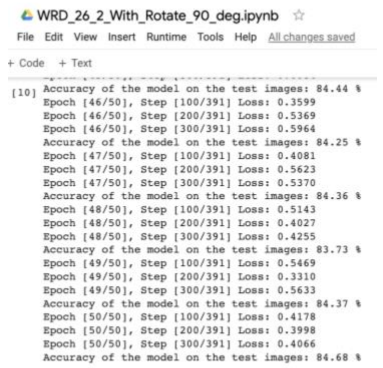


```
Epoch [46/50], Step [300/391] Loss: 0.1273
Accuracy of the model on the test images: 92.08 %
Epoch [47/50], Step [100/391] Loss: 0.0612
Epoch [47/50], Step [200/391] Loss: 0.1060
Epoch [47/50], Step [300/391] Loss: 0.1342
Accuracy of the model on the test images: 91.9 %
Epoch [48/50], Step [100/391] Loss: 0.0905
Epoch [48/50], Step [200/391] Loss: 0.1720
Epoch [48/50], Step [300/391] Loss: 0.0493
Accuracy of the model on the test images: 91.84 %
Epoch [49/50], Step [100/391] Loss: 0.1206
Epoch [49/50], Step [200/391] Loss: 0.1156
Epoch [49/50], Step [300/391] Loss: 0.1035
Accuracy of the model on the test images: 92.04 %
Epoch [50/50], Step [100/391] Loss: 0.1474
Epoch [50/50], Step [200/391] Loss: 0.1009
Epoch [50/50], Step [300/391] Loss: 0.0630
```

Figure 5: WRN-26-2 with Dropout & Data Augmentation (# Params: 4.32M)

We tried using other augmentation techniques as well, one of which is rotation (90 degree). However, rotation didn't optimize our model and accuracy was reduced.




WRD_26_2_With_Rotate_90_deg.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text

```
[10] Accuracy of the model on the test images: 84.44 %
     Epoch [46/50], Step [100/391] Loss: 0.3599
     Epoch [46/50], Step [200/391] Loss: 0.5369
     Epoch [46/50], Step [300/391] Loss: 0.5964
     Accuracy of the model on the test images: 84.25 %
     Epoch [47/50], Step [100/391] Loss: 0.4081
     Epoch [47/50], Step [200/391] Loss: 0.5623
     Epoch [47/50], Step [300/391] Loss: 0.5370
     Accuracy of the model on the test images: 84.36 %
     Epoch [48/50], Step [100/391] Loss: 0.5143
     Epoch [48/50], Step [200/391] Loss: 0.4027
     Epoch [48/50], Step [300/391] Loss: 0.4255
     Accuracy of the model on the test images: 83.73 %
     Epoch [49/50], Step [100/391] Loss: 0.5469
     Epoch [49/50], Step [200/391] Loss: 0.3310
     Epoch [49/50], Step [300/391] Loss: 0.5633
     Accuracy of the model on the test images: 84.37 %
     Epoch [50/50], Step [100/391] Loss: 0.4178
     Epoch [50/50], Step [200/391] Loss: 0.3998
     Epoch [50/50], Step [300/391] Loss: 0.4066
     Accuracy of the model on the test images: 84.68 %
```

Figure 6: WRN-26-2 with Dropout & Data Augmentation (with Rotation) (# Params: 4.32M)

Model Scaling (can be scaled for different resource constraints [5]) is another widely practiced technique for augmentation and referring to the work of Mingxing et al. [4], we make use of compound scaling method:

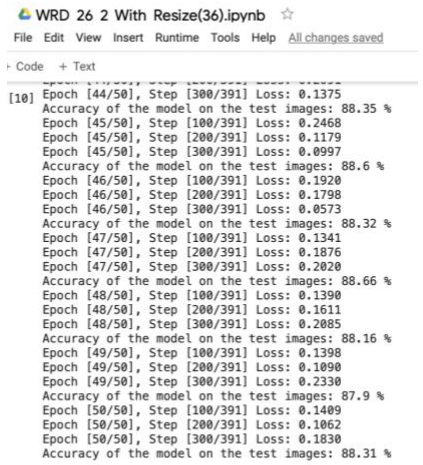
$$depth: d = \alpha^{\varphi}$$

$$width: w = \beta^{\varphi}$$

$$resolution: r = \gamma^{\varphi}$$

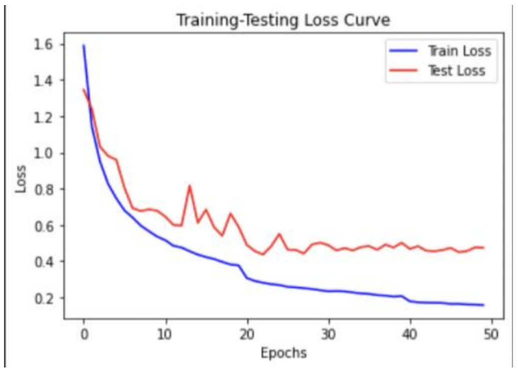
$$\exists\ \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2;\ \alpha, \beta, \gamma \geq 1$$

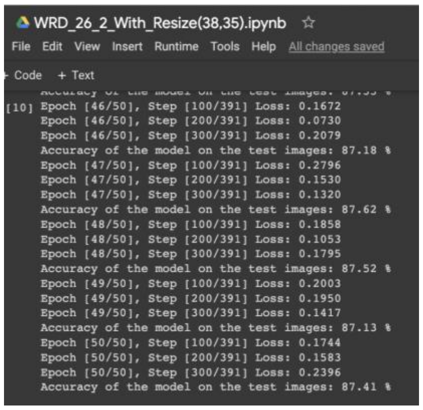
We, then, tried using transformation techniques, and experimented with RandomAffine (Random affine transformation of the image keeping center invariant) and ColorJitter (Randomly change the brightness, contrast, and saturation of an image). A note on data augmentation - Data Augmentation approaches overfitting from the root of the problem, the training dataset, as observed in the work by Shorten et al. [3].


WRD 26 2 With Resize(36).ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text

```
[10] Epoch [44/50], Step [300/391] Loss: 0.1375
     Accuracy of the model on the test images: 88.35 %
     Epoch [45/50], Step [100/391] Loss: 0.2468
     Epoch [45/50], Step [200/391] Loss: 0.1179
     Epoch [45/50], Step [300/391] Loss: 0.0997
     Accuracy of the model on the test images: 88.6 %
     Epoch [46/50], Step [100/391] Loss: 0.1920
     Epoch [46/50], Step [200/391] Loss: 0.1798
     Epoch [46/50], Step [300/391] Loss: 0.0573
     Accuracy of the model on the test images: 88.32 %
     Epoch [47/50], Step [100/391] Loss: 0.1341
     Epoch [47/50], Step [200/391] Loss: 0.1876
     Epoch [47/50], Step [300/391] Loss: 0.2020
     Accuracy of the model on the test images: 88.66 %
     Epoch [48/50], Step [100/391] Loss: 0.1390
     Epoch [48/50], Step [200/391] Loss: 0.1611
     Epoch [48/50], Step [300/391] Loss: 0.2085
     Accuracy of the model on the test images: 88.16 %
     Epoch [49/50], Step [100/391] Loss: 0.1398
     Epoch [49/50], Step [200/391] Loss: 0.1090
     Epoch [49/50], Step [300/391] Loss: 0.2330
     Accuracy of the model on the test images: 87.9 %
     Epoch [50/50], Step [100/391] Loss: 0.1409
     Epoch [50/50], Step [200/391] Loss: 0.1062
     Epoch [50/50], Step [300/391] Loss: 0.1830
     Accuracy of the model on the test images: 88.31 %
```

Figure 7: WRN-26-2 with Dropout & Data Augmentation (with Resize 36, 36) (# Params: 4.32M)





```
     Accuracy of the model on the test images: [illegible] %
[10] Epoch [46/50], Step [100/391] Loss: 0.1672
     Epoch [46/50], Step [200/391] Loss: 0.0730
     Epoch [46/50], Step [300/391] Loss: 0.2079
     Accuracy of the model on the test images: 87.18 %
     Epoch [47/50], Step [100/391] Loss: 0.2796
     Epoch [47/50], Step [200/391] Loss: 0.1530
     Epoch [47/50], Step [300/391] Loss: 0.1320
     Accuracy of the model on the test images: 87.62 %
     Epoch [48/50], Step [100/391] Loss: 0.1858
     Epoch [48/50], Step [200/391] Loss: 0.1053
     Epoch [48/50], Step [300/391] Loss: 0.1795
     Accuracy of the model on the test images: 87.52 %
     Epoch [49/50], Step [100/391] Loss: 0.2003
     Epoch [49/50], Step [200/391] Loss: 0.1950
     Epoch [49/50], Step [300/391] Loss: 0.1417
     Accuracy of the model on the test images: 87.13 %
     Epoch [50/50], Step [100/391] Loss: 0.1744
     Epoch [50/50], Step [200/391] Loss: 0.1583
     Epoch [50/50], Step [300/391] Loss: 0.2396
     Accuracy of the model on the test images: 87.41 %
```

Figure 8: WRN-26-2 with Dropout & Data Augmentation (with Resize 38, 35) (# Params: 4.32M)

Adam was the optimizer used in all previously mentioned models. RMSPROP, ADAM, and N-ADAM are found to never underperform SGD, NESTEROV, or MOMENTUM [8].

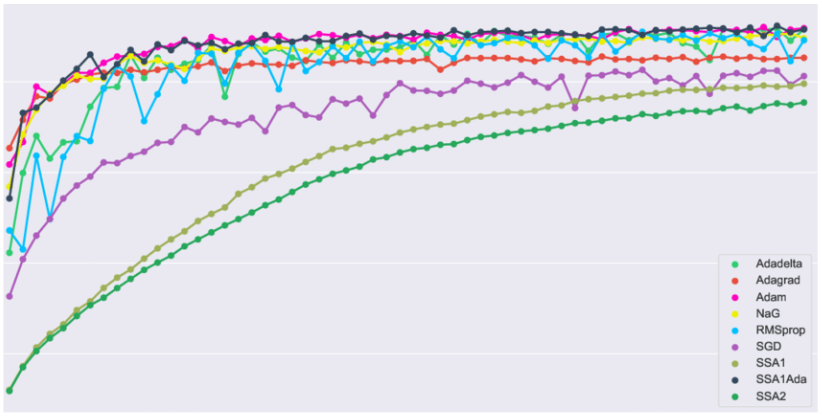
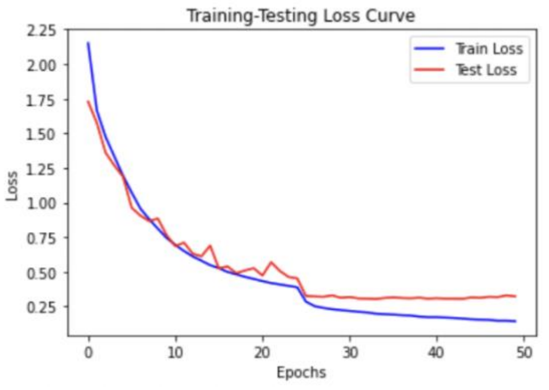
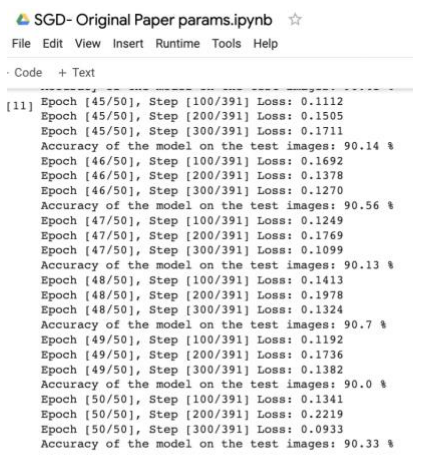


Figure 9: Accuracy Comparison for Different Optimizers on test CIFAR10 Dataset [9]

We tried using Adadelta optimizer but with worse results. Our next choice was Stochastic Gradient Descent optimized with weight decay of 0.0001 and momentum of 0.9 and starting learning rate 0.1, as suggested in [5], and performance did not enhance over Adam. Using Adam along with SGD as a Meta optimizer has been proposed [6], but its application was too convoluted for this project.

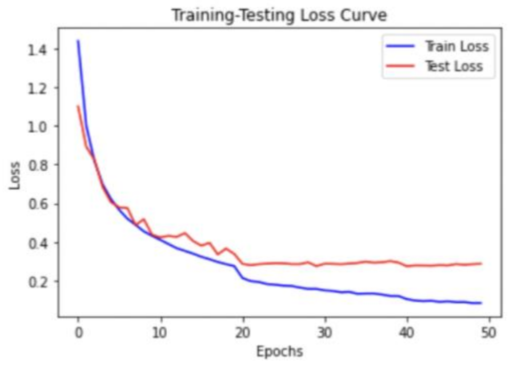


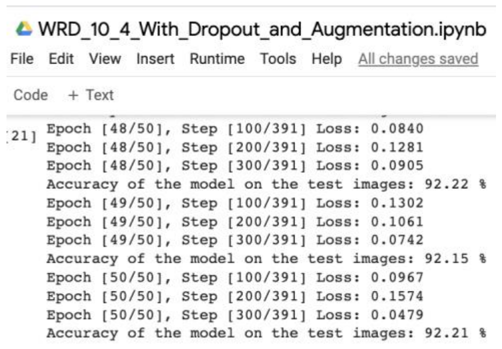

SGD- Original Paper params.ipynb
File Edit View Insert Runtime Tools Help
Code + Text

```
[11] Epoch [45/50], Step [100/391] Loss: 0.1112
     Epoch [45/50], Step [200/391] Loss: 0.1505
     Epoch [45/50], Step [300/391] Loss: 0.1711
     Accuracy of the model on the test images: 90.14 %
     Epoch [46/50], Step [100/391] Loss: 0.1692
     Epoch [46/50], Step [200/391] Loss: 0.1378
     Epoch [46/50], Step [300/391] Loss: 0.1270
     Accuracy of the model on the test images: 90.56 %
     Epoch [47/50], Step [100/391] Loss: 0.1249
     Epoch [47/50], Step [200/391] Loss: 0.1769
     Epoch [47/50], Step [300/391] Loss: 0.1099
     Accuracy of the model on the test images: 90.13 %
     Epoch [48/50], Step [100/391] Loss: 0.1413
     Epoch [48/50], Step [200/391] Loss: 0.1978
     Epoch [48/50], Step [300/391] Loss: 0.1324
     Accuracy of the model on the test images: 90.7 %
     Epoch [49/50], Step [100/391] Loss: 0.1192
     Epoch [49/50], Step [200/391] Loss: 0.1736
     Epoch [49/50], Step [300/391] Loss: 0.1382
     Accuracy of the model on the test images: 90.0 %
     Epoch [50/50], Step [100/391] Loss: 0.1341
     Epoch [50/50], Step [200/391] Loss: 0.2219
     Epoch [50/50], Step [300/391] Loss: 0.0933
     Accuracy of the model on the test images: 90.33 %
```

Figure 10: WRN-26-2 with SGD Optimizer (# Params: 4.32M)

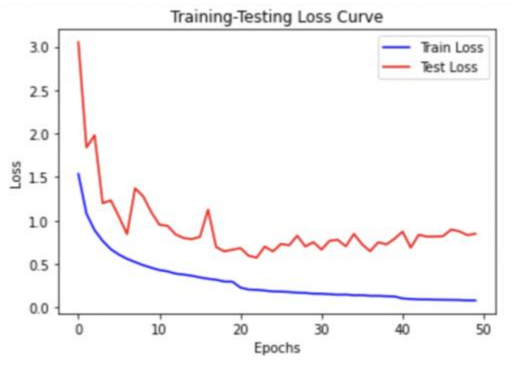
Going forward with our model, we widen our model (widening factor 4) and increase the number of parameters while keeping the limit capped at 5 million. We have a total of 10 layers, 4 residual layers with one residual block, each block with 2 convolutional layers and a dropout. This model of ours has the best accuracy so far. To increase the number of parameters further, we set the widening factor to 2 and the number of layers to 32. We changed the parameters for normalization, but the results did not match our expectations and did not better our 10-layer 4-widening factor model.

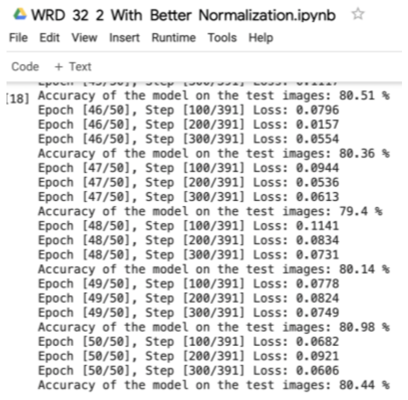



WRD_10_4_With_Dropout_and_Augmentation.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Code + Text

```
[21] Epoch [48/50], Step [100/391] Loss: 0.0840
     Epoch [48/50], Step [200/391] Loss: 0.1281
     Epoch [48/50], Step [300/391] Loss: 0.0905
     Accuracy of the model on the test images: 92.22 %
     Epoch [49/50], Step [100/391] Loss: 0.1302
     Epoch [49/50], Step [200/391] Loss: 0.1061
     Epoch [49/50], Step [300/391] Loss: 0.0742
     Accuracy of the model on the test images: 92.15 %
     Epoch [50/50], Step [100/391] Loss: 0.0967
     Epoch [50/50], Step [200/391] Loss: 0.1574
     Epoch [50/50], Step [300/391] Loss: 0.0479
     Accuracy of the model on the test images: 92.21 %
```

Figure 11: WRN-10-4 with Dropout & Data Augmentation (# Params: 4.73M)




WRD 32 2 With Better Normalization.ipynb
File Edit View Insert Runtime Tools Help
Code + Text

```
[18] Accuracy of the model on the test images: 80.51 %
     Epoch [46/50], Step [100/391] Loss: 0.0796
     Epoch [46/50], Step [200/391] Loss: 0.0157
     Epoch [46/50], Step [300/391] Loss: 0.0554
     Accuracy of the model on the test images: 80.36 %
     Epoch [47/50], Step [100/391] Loss: 0.0944
     Epoch [47/50], Step [200/391] Loss: 0.0536
     Epoch [47/50], Step [300/391] Loss: 0.0613
     Accuracy of the model on the test images: 79.4 %
     Epoch [48/50], Step [100/391] Loss: 0.1141
     Epoch [48/50], Step [200/391] Loss: 0.0834
     Epoch [48/50], Step [300/391] Loss: 0.0731
     Accuracy of the model on the test images: 80.14 %
     Epoch [49/50], Step [100/391] Loss: 0.0778
     Epoch [49/50], Step [200/391] Loss: 0.0824
     Epoch [49/50], Step [300/391] Loss: 0.0749
     Accuracy of the model on the test images: 80.98 %
     Epoch [50/50], Step [100/391] Loss: 0.0682
     Epoch [50/50], Step [200/391] Loss: 0.0921
     Epoch [50/50], Step [300/391] Loss: 0.0606
     Accuracy of the model on the test images: 80.44 %
```

Figure 12: WRN-32-2 with Dropout & Data Augmentation (# Params: 4.98M)

With our 10-layer, 4-widening factor, we try different epoch cycles – starting with 40 epochs, all the way till 200 epochs. Accuracy as measured with different epoch cycles is shown in the results section.

Our final architecture, thus, has a total of 10 layers, consisting of 4 residual layers of 1 block each, each block has 2 convolutional layers with batch normalization and ReLU activation, and with a dropout layer between them. After 4 residual layers, the output goes through average pooling and a fully connected layer before the final result.

The implementation for the above architecture can be found at https://github.com/kunalkashyap855/wide-resnet-cifar10. The code is built upon the basic Deep ResNet code authored by Yerlan Idelbayev [7].

3 Results

A list of all exhaustive experimentations performed over the course of this project over various ResNet architectures, data augmentation techniques, optimizers, and epoch cycling:

Table 1: Accuracy and Params of all models

Architecture	**Details**	**# Layers**	**# Params**	**Epochs**	**Accuracy**
ResNet110		110	1.73M	50	85.49
WRN-26-2	Without Dropout and Augmentation	26	4.32M	50	86.25
WRN-26-2	With Dropout without Augmentation			50	87.97
WRN-26-2	With Dropout and Augmentation (Horizontal Flip and Crop)			40	91.52
				50	92.04
				200	92.27
WRN-26-2	With Rotation (90) Transform			50	84.69
WRN-26-2	With Resize (36, 36) Transform			50	88.31
WRN-26-2	With Resize (38, 35) Transform			50	87.41
WRN-26-2	With Random Affine and Color Jitter Tranform			50	91.69
WRN-26-2	SGD used as optimizer			50	90.33
				200	91.55
WRN-10-4	With Dropout and Augmentation (Horizontal Flip and Crop)	10	4.73M	50	92.21
				200	**92.37**

WRN-32-2	With Dropout and Augmentation (Horizontal Flip and Crop)	32	4.98M	50	80.44

4 Conclusion

We conclude that Wide Residual Networks offer more efficient training and higher test accuracy than Deep ResNets on the CIFAR-10 dataset. Adding a Dropout layer helps in reducing the train and validation error, but not to a significant effect. We also conclude that various data augmentation techniques such as Horizontal Flipping, Cropping and Color Jittering helps in reducing overfitting and hence reduces the validation error. Finally, using the Adam optimizer with Wide ResNets gives better results than using Stochastic Gradient Descent.

References

[1] Glorot, Xavier & Bengio, Yoshua (2010) Understanding the Difficulty of Training Deep Feed-forward Neural Networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 249-256. Cambridge, MA: MIT Press.

[2] Zagoruyko, Sergey & Komodakis, Nikos (2016) Wide Residual Networks. In Richard C. Wilson, Edwin R. Hancock and William A. P. Smith, editors, *Proceedings of the British Machine Vision Conference (BMVC)*, pp. 87.1-87.12. BMVA Press.

[3] Shorten, Connor & Khoshgoftaar, Taghi M. (2019) A Survey on Image Data Augmentation for Deep Learning. In *Journal of Big Data*.

[4] Tan, Mingxing & Le, Quoc V. (2019) EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In *International Conference on Machine Learning*.

[5] He, Kaiming & Zhang, Xiangyu & Ren, Shaoqing & Sun, Jian (2016) Deep Residual Learning for Image Recognition. In Richard C. Wilson, Edwin R. Hancock and William A. P. Smith, editors, *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Las Vegas, NV.

[6] Landro, Nicola & Gallo, Ignazio & La Grassa, Riccardo (2021) Combining Optimization Methods Using an Adaptive Meta Optimizer. In Akemi Galvez Tomida, editor, *MDPI Algorithms*.

[7] Idelbayev, Yerlan ResNet CIFAR-10 Pytorch Code. University of California, Merced, CA. (https://github.com/akamaster/pytorch_resnet_cifar10/blob/master/resnet.py)

[8] Pinta, Titus & Boros, Imre & Alecsa, Christian Daniel (2019) New Optimization Algorithms for Neural Network Training using Operator Splitting Techniques. In *Neural Networks*.

[8] Choi Dami, & J. Shallue, Christopher & Nado, Zachary & Lee, Jaehoon & J. Maddison, Chris & E. Dahl George (2020) On Empirical Comparisons of Optimizers of Deep Learning. In *International Conference on Learning Representations ICLR-2020*.

[10] Huang, Gao & Sun, Yu & Liu, Zhuang & Daniel, Sedra & Q. Weinberger, Kilian (2016) Deep Networks with Stochastic Depth. In Bastian Leibe, Jiri Matas, Nicu Sebe and Max Welling, editors, *Computer Vision – ECCV*, pp. 646-661.