Mini-Project 1: Residual Network Design

Xuanzhou Chen
Department of Electrical & Computer Engineering
New York University
Brooklyn, NY 11201
xc2425@nyu.edu

Abstract

RetNet has been a popular network since it came out, many improvements and derivation has been made on this model during the past years. In this project, we explored the best performance of ResNet model on CIFAR 10 dataset using around 4.9M total training paramters. Different groups of approaches including model structural modifications, data augmentations, optimizers, regularization schemes, learning rates schedules, batch size and epoch settings were adopted, investigated, recorded and compared to find the best model with maximal test accuracy. Github Link: https://github.com/xchen793/DL_mini_ResNet

1 Introduction

Resnet was first proposed by He et al. [2016]in 2015, a CNN model as an effective technique for improving the accuracy of modern image classification. He et al. [2016] suggested it as a solution to the degradation problem, and proved that the ResNet boosts the performance in deeper layers in the experiments on ImageNet and CIFAR-10 classifications.

Compared to the plain network, the magic of the ResNet lies in the construction of the identity mapping (i.e. shortcut connections). According to He et al. [2016], identity mapping layers $\mathbf{x}$ are added to the shallower architecture $\mathcal{F}(\mathbf{x})$ to build a residual block. He et al. [2016] further presented an 18-layer ResNet model based on the stacked residual blocks (see Fig. 1). In each residual block, the desired underlying mapping is a operation by a shortcut connection and element-wise addition, formally denoted as $\mathcal{H}(\mathbf{x}) = \mathcal{F}(\mathbf{x}) + \mathbf{x}$.

Our implementation for ResNet in this project follows this basic architecture. Each residual block contains two convolutional layers with a skip connection from the block's input to the block's output. F is the following sequence of operations: conv $\rightarrow$ bn $\rightarrow$ bn $\rightarrow$ relu $\rightarrow$ conv $\rightarrow$ bn. $\mathcal{H}(\mathbf{x})$ is implemented as relu(F(x) + conv(x)), where conv(x) corresponds to the skip connection. The ResNet consists of N residual layers. Each residual layer contains one or more residual blocks. Residual layers are preceded by a fixed convolutional block and followed by an average pooling and fully connected layer.

The ResNet architecture (see Fig. 2) by design was aimed to maximize accuracy on the CIFAR-10 dataset under 5M trainable parameters. The hyperparameters we could change are the number of residual layers, N, the number of residual blocks in the residual layer i, B_i, the number of channels in residual layer i, C_i, the convolutional kernel size in residual layer i, F_i, the skip connection kernel size in residual layer i, K_i, the average pool kernel size, P.

Data augmentation strategies, choices of optimizers, regularization schemes, multiple sets of batch size and epochs of training are also adopted to enhance the training procedure and to improve the model performance. Test accuracies, training accuracies and training loss at a certain epoch in different groups are recorded and compared by control group design for the evaluation.

Preprint. Under review.

2 Methodology

Our architecture search process was based on fine tuning the hyperparameters in the model, selecting data augmentation methods, choosing different optimizers, using different regularization schemes, adopting multiple learning rate schedulers.

2.1 Model - Hyperparameters Fine-tuning

Inspired by the original paper of Resnet, where He et al. [2016] argued that the Resnet has better performance as the networks dives deeper, we tried different number of residual layers: Group 1 - 104 layers, Group 2 - 30 layers, Group 3 - 30 layers, Group 4 - 38 layers, Group 5 - 24 layers. Gaining the inspiration from the structure of the wide residual networks built by Zagoruyko and Komodakis [2016] and the depth scaling & width scaling from the Efficient Net created by Tan and Le [2019], We made the layers deeper by reducing the number of the input channels. Different kernel size in the average pooling were also compared (see Group 2 & Group 3).

For the experiment, we set the learning rate to 0.9, Batch size to 128, the number of epochs to 50. We adjust the learning rate using decay rate(=0.9) every 5 epochs. We chose the optimizer as Adam (using L2 regularization, weight decay = $1e^{-6}$). Data augmentation included.

Group	B_i	C_i	F_i	K_i	P	Num of Param
1	[9,15,15,12]	[16,32,64,128]	[3,3,3,3]	[1,1,1,1]	4	4,891,482
2	[3,4,4,3]	[32,64,128,256]	[3,3,3,3]	[1,1,1,1]	4	4,735,658
3	[3,4,4,3]	[32,64,128,256]	[3,3,3,3]	[1,1,1,1]	3	4,735,658
4	[4,5,5,4]	[28,56,112,224]	[3,3,3,3]	[1,1,1,1]	4	4,828,134
5	[3,2,2,2,2]	[21,42,84,168,336]	[3,3,3,3,3]	[1,1,1,1,1]	2	4,832,425

Group	Test Accuracy(%)	Training Accuracy(%)	Loss
1	90.910	99.346	0.019
2	92.040	99.545	0.013
3	92.160	99.650	0.011
4	92.020	99.626	0.012
5	90.440	99.458	0.016

1 Testing accuracy and training accuracy and trainng loss on **CIFAR-10** dataset.

2 All methods are with data augmentation.

3 Batch size = 128. Total epoch = 50

By comparison, we choose Group 3 for the model hyperparameters setting, since it gives us lowest loss and highest accuracies.

2.2 Data Augmentation

As Shijie et al. [2017]pointed out in his paper, the combination of data augmentation works better than using one single data augmentation methods. We combined different data augmentation strategies, including random erasing, random flipping, random cropping, and normalization, to see if such a combination could boost the model performance.

First, We adopted the Random Erasing, which is a data augmentation strategy proposed by Zhong et al. [2017] The base setting (probability $p = 0.5$, the area ratio $s_h = 0.4$, and the aspect ratio $r_1 = 0.3$) in the paper was used here. Then we cropped the images and flipped them horizontally to enlarge the dataset, inspired by Martin et al. [2019]. At last, we normalized the image dataset.

Random erasing (p = 0.5, sh = 0.4, r1 = 0.3) + Random Cropping + Random Flipping vs. Random Cropping + Random Flipping was compared using Adam optimizer. (see Fig.2 & Fig.3 below) The experiment shows that using random cropping and random flipping is necessary, it significantly improves the test accuracy by 5.36%. However, adding random erasing alone to random cropping and random flipping does not effectively lower the accuracies.

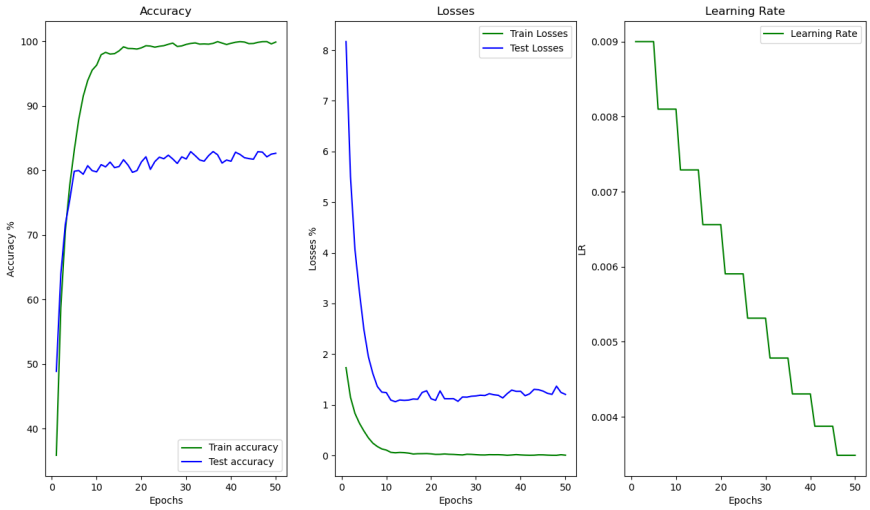

Figure 1: Clean (no DataAug) Test accuracy = 82.650%. Training accuracy = 99.878%. Loss = 0.004

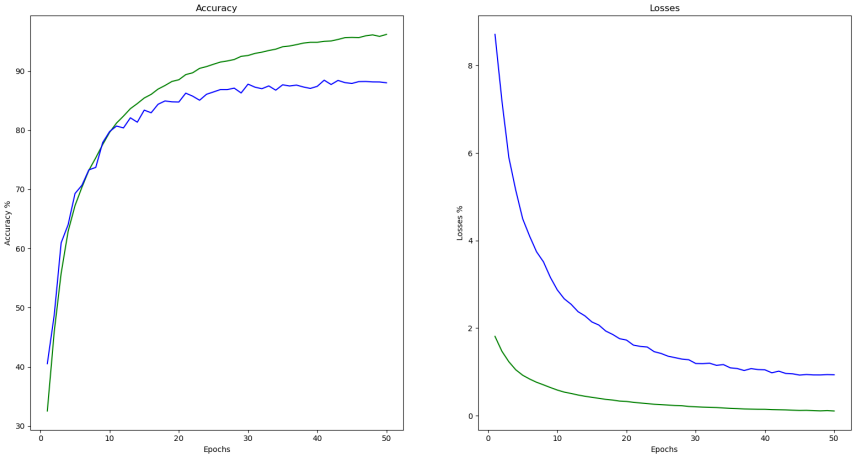

Figure 2: Without Random Erasing. Test accuracy = 88.010%. Training accuracy = 96.192%. Loss = 0.107

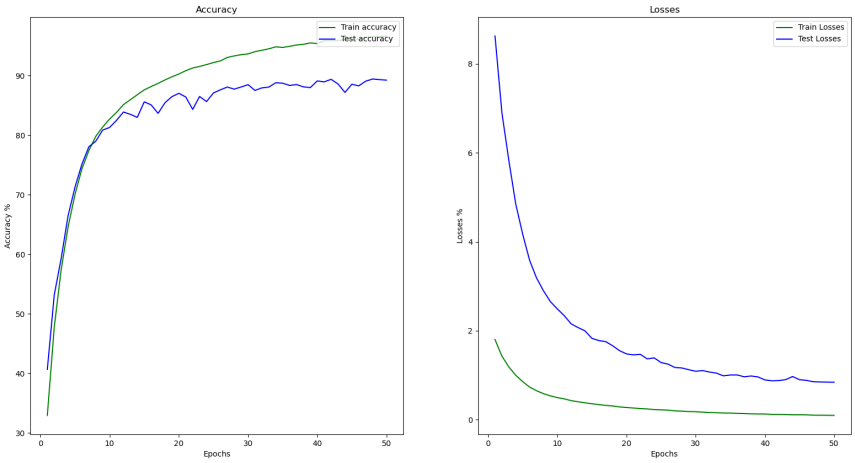

Figure 3: With Random Erasing. Test accuracy = 88.450%. Training accuracy = 94.842%. Loss = 0.147

By comparison, we choose random flipping + random cropping as the data augmentation strategy.

2.3 Optimizer

Selecting an optimizer is a central step in the contemporary deep learning pipeline,Choi et al. [2019] found that RMSPROP, ADAM,and NADAM never underperformed SGD, NESTEROV, or MOMENTUM under our most exhaustive tuning protocol Bengio [2012]. The model we adopted for the Learning rate test is Group 3 (B_i = [3,4,4,3], P = 3]), and epoch number was set to 50. Learning rate was initialized as 0.01, with decay rate = 0.9. Optimizers such as **SGD**, **SGD + momentum**, **Adam** Kingma and Ba [2014] and **RMSprop** with different α's were tested. Data augmentation (i.e., the combination of random erasing(p = 0.5, sh = 0.4, r1 = 0.3), random flipping, and random horizontal cropping) is used in all experiments.

Group	Optimizer	momentum	α (RMSprop)	(β_1, β_2) (Adam)
1	SGD	0	/	/
2	SGD + momentum	0.9	/	/
3	Adam	0	/	(0.9, 0.999)
4	RMSprop	0	0.99	/
5	RMSprop	0	0.9	/

Group	Test Accuracy(%)	Training Accuracy(%)	Loss
1	87.680	95.874	0.118
2	91.330	94.190	0.163
3	91.000	97.880	0.061
4	68.090	67.964	0.893
5	70.980	76.477	0.678

1 Testing accuracy and training accuracy and training loss on **CIFAR-10** dataset.

2 All methods are with data augmentation.

3 Batch size = 128. Total epoch = 50.

By comparison, we choose the Adam optimizer with (0.9, 0.99).

2.4 Regularization

The reason why we tried to use L2 norm is because in the training process, weight set is not necessarily unique: there might be many similar W that correctly classify the examples. We wish to encode some preference for a certain set of weights W over others to remove this ambiguity. We can do so by extending the loss function with a regularization penalty R(W). The most common regularization penalty is the squared L2 (Li).

Dropout has been used by many successful architectures, different from the experiments Zagoruyko and Komodakis [2016] did in their paper, we only used the dropout in the dense layer to see if it can carry out some new state-of-art results.

The model we adopted for the regularization scheme test is Group 8 (5-layer), we set the initialized learning rate to 0.01, batch size to 128, the number of epochs to 100. We adjust the learning rate using decay rate(= 0.9) every 5 epochs. We chose the optimizer as Adam in this experiment. Note that the dropout only added in the dense layer.

Group	Regularization Methods	Weight Decay	Dropout Rate
1	L1	$1e^{-10}$	/
2	L1	$1e^{-6}$	/
3	L2	$1e^{-10}$	/
4	L2	$1e^{-6}$	/
5	Dropout	/	0.4
6	Dropout	/	0.5
7	Dropout	/	0.6

Group	Test Accuracy(%)	Training Accuracy(%)	Loss
1	89.490	96.900	0.087
2	89.230	95.582	0.305
3	89.720	97.190	0.081
4	89.290	94.228	0.165
5	89.670	96.588	0.095
6	90.220	96.366	0.103
7	89.420	96.132	0.111

1 Testing accuracy and training accuracy and trainng loss on **CIFAR-10** dataset.

2 All methods are with data augmentation.

3 Batch size = 128. Total epoch = 100.

By comparison, we choose the L2 with weight decay = 1e-10 as regularization schemes.

2.5 Learning Rate

For the learning rate, we used a learning rate scheduler which automatically adjusts the learning rate during training epochs.

In Farooq and Hafeez [2020] work, they provided a cyclical learning rate solution to periodically change the learning rate. The model we adopted for the Learning rate test is Group 8 (5-layer). Epoch number was set to 50, and batch size is set to 128. Learning rate was initialized as 0.01. Optimizers such as SGD with momentum = 0.9, weight decay = 5e-4, Adam(β = (0.9, 0.999), ϵ = 1e-08, weight decay = 1e-10) and RMSprop ($\alpha = 0.99$, $\epsilon = 1e-8$, weight decay $= 5e-4$) were tested on different sets of decay rate of the learning rate (0.95, 0.9, 0.8), and update per epoch settings (e.g. Epochs/update = 5 means the learning rate is updated every 5 epochs). Data augmentation (i.e., the combination of random erasing(p = 0.5, sh = 0.4, r1 = 0.3), random flipping, and random horizontal cropping) is used in all experiments.

Group	Optimizer	Decay Rate(LR)	Epochs/update
1	SGD	0.95	5
2	SGD	0.9	5
3	SGD	0.8	5
4	SGD	0.9	10
5	Adam	0.9	5
6	Adam	0.8	5
7	RMSProp	0.9	5

Group	Test Accuracy(%)	Training Accuracy(%)	Loss
1	89.400	95.332	0.132
2	89.930	96.404	0.103
3	89.310	97.306	0.078
4	88.470	95.124	0.133
5	89.190	97.114	0.081
6	90.220	98.542	0.041
7	89.420	96.132	0.111

1 Testing accuracy and training accuracy and training loss on **CIFAR-10** dataset.

2 All methods are with data augmentation.

3 Batch size = 128. Total epoch = 50

By comparison, if we choose Adam as optimizer, we would use 0.8 as the decay rate of LR and update it every 5 epochs.If we choose SGD as optimizer, we would adopt 0.8 as the decay rate and set 5 epochs per update. The performance of RMSprop is our optimal choice, and its performance is slightly worse than the Adam and SGD.

3 Results

From above experiments we have conducted so far, we define our final model as following:

1. For the model hyperparameters, we choose $B_i = [3, 4, 4, 3]$, $C_i = [32, 64, 128, 256]$, $F_i = [3, 3, 3, 3]$, $K_i = [1, 1, 1, 1]$, $P = 3$, number of parameters = 4,735,658.

2. For the data augmentation, we choose random horizontal flipping and random cropping.

3. For the optimizer, we choose Adam.

4. For the regularization scheme, we choose the L2 with weight decay = 1e-10.

5. For the learning rate, since we choose Adam as optimizer, we would use 0.8 as the decay rate of LR and update it every 5 epochs.

Since the more epochs we had, the higher accuracies we got, so we set the total training epochs as 150.

The result is shown in Figure 4 at epoch 150: Test accuracy = 91.980 %, Training accuracy = 99.984%. Training Loss = 0.001. But the highest test accuracy could achieve at 92.040 % at epoch 144.

4 Conclusion

In this project, we went though ResNet 18 residual network model and investigated the model training and testing accuracy and loss on CIFAR10 dataset. We demonstrated the test and train accuracies after tuning models with different hyperparameters. In addition, we explored several data augmentation strategies, optimizers, regularization schemes and learning rate schedulers for further model improvements. As a result, we found that our final model achieves the best accuracy on CIFAR10 testset based our experiments at 92.040 %.

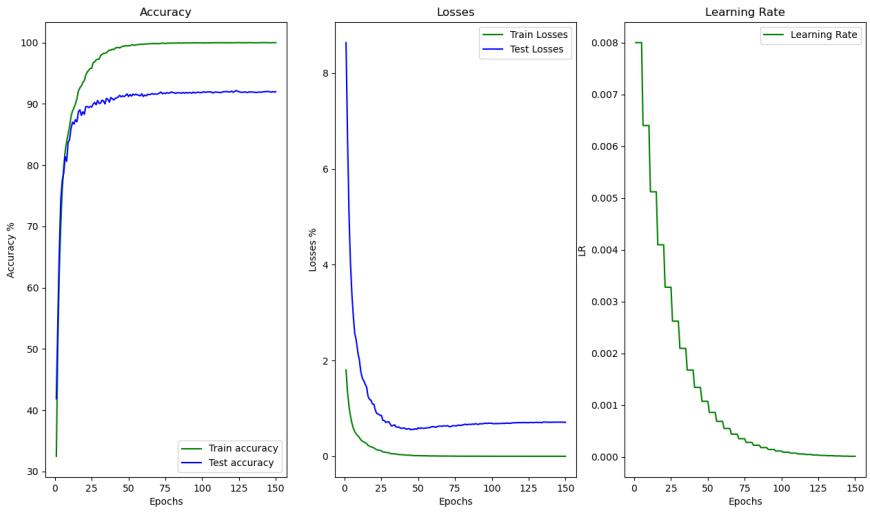

Figure 4: Accuracies & Loss vs. Epochs

References

Yoshua Bengio. Practical recommendations for gradient-based training of deep architectures. In *Neural networks: Tricks of the trade*, pages 437–478. Springer, 2012.

Dami Choi, Christopher J Shallue, Zachary Nado, Jaehoon Lee, Chris J Maddison, and George E Dahl. On empirical comparisons of optimizers for deep learning. *arXiv preprint arXiv:1910.05446*, 2019.

Muhammad Farooq and Abdul Hafeez. Covid-resnet: A deep learning framework for screening of covid19 from radiographs. *arXiv preprint arXiv:2003.14395*, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014. URL `https://arxiv.org/abs/1412.6980`.

Feifei Li. Cs231n course note. URL `http://cs231n.stanford.edu/`.

Matthieu Martin, Bruno Sciolla, Michaël Sdika, Philippe Quétin, and Philippe Delachartre. Segmentation of neonates cerebral ventricles with 2d cnn in 3d us data: suitable training-set size and data augmentation strategies. In *2019 IEEE International Ultrasonics Symposium (IUS)*, pages 2122–2125, 2019. doi: 10.1109/ULTSYM.2019.8925799.

Jia Shijie, Wang Ping, Jia Peiyi, and Hu Siping. Research on data augmentation for image classification based on convolution neural networks. In *2017 Chinese automation congress (CAC)*, pages 4165–4170. IEEE, 2017.

Mingxing Tan and Quoc Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6105–6114. PMLR, 09–15 Jun 2019. URL `https://proceedings.mlr.press/v97/tan19a.html`.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks, 2016. URL `https://arxiv.org/abs/1605.07146`.

Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation, 2017. URL `https://arxiv.org/abs/1708.04896`.