A modified ResNet Architecture with Optimal Test Accuracy

Sayam Dhingra ,Yami Naik, Bhautik Sudani

New York University,
New York, USA.

Abstract

The Residual Networks (ResNet) were created to combine the benefits of having a deeper neural network while also addressing the issue of vanishing gradient. They do this by using skip connections, which skip certain layers in the network and pass the output of one layer as input to the next. This allows gradients to flow directly from later layers to earlier layers during backpropagation, which helps to alleviate the problem of vanishing gradients. ResNet18 is one of the ResNet models and consists of 18 layers and around 11 million trainable parameters. In this report, the goal is to maximize the test accuracy of the ResNet18 model while ensuring that the number of trainable parameters in any modified architecture does not exceed 5 million. The ResNet18 model is experimented with by altering its layer structure and using various parameters and optimizers. Based on the experimental results, modified versions of the ResNet18 architecture are proposed that exhibit the highest reliable test accuracy within the given constraint. The codebase for this work can be found at https://github.com/yaminaik/OptimizedResNet18.

Introduction

Convolutional Neural Networks (CNNs) are widely used in deep learning for computer vision tasks due to their ability to capture important features. Although Transformers have been introduced as an alternative, CNNs remain more popular since they require less data for training. This is because CNNs have inherent priors that enable them to learn quickly, while Transformers must learn these priors during training. Researchers originally believed that deeper networks would yield better results because they can learn more complex features. However, they discovered that beyond a certain point, the accuracy of the network saturates and even decreases. This is due to the vanishing gradient problem that occurs in Deep Neural Networks.

Residual Networks (ResNets) tackle the problem of vanishing gradients by utilizing skip connections. These skip connections allow gradients to flow directly from later layers to earlier layers during backpropagation, bypassing the intermediate layers that can lead to vanishing gradients. Additionally, skip connections enable the network to learn features corresponding to lower-level semantic information extracted from the input, preventing the loss of important information. In this study, we modified the ResNet18 architecture by adjusting input parameters, layer structure, inducing dropouts, and other techniques to maximize test accuracy while limiting the number of parameters to 5 million. Our proposed architecture is a variant of ResNet18 that achieves maximum accuracy within the parameter constraint. The original ResNet18 has 18 deep layers, 72-layer architecture, and around 11 million parameters.(Web References) (Pablo Ruiz 2018) (Nikolas Adaloglou 2020) (Great Learning 2022)

The 72-layer architecture of the ResNet18 model contains 18 deep layers that contain approximately 11 million parameters. In this study, we tweak the input parameters, alter the layer structure, induce dropouts, and other approaches to create an architecture that demonstrates maximum test accuracy using the ResNet18 base architecture. The test accuracy is maximized given the parameter constraint in the suggested architecture, which is a modified version of ResNet18. All of our trials were carried out while keeping in mind the restriction that the new architecture cannot include more than 5 million parameters.

Methodology

Modifying the Architectural Layer

In order to decrease the number of parameters in our model to below 5 million, we made modifications to the ResNet-18 architecture.
There were two methods we used to achieve this goal:

Reducing the number of channels. (Table 1)
Reducing the number of layers in the model
For example, instead of using 64, 128, 256, and 512 channels, we reduced it to 32, 64, 128, and 256 channels which resulted in approximately 2.8 million parameters.

Our main objective was to avoid overfitting during training and maximize the test accuracy to ensure a good model. We aimed to keep the test accuracy as close as possible to the training accuracy.

Layer Structure	Parameters (in M)
[2,2,2,2]	11.1
[2,2,2,1]	6.45
[2,1,2,1]	6.15
[2,1,1,1]	4.9
[1,1,1,2]	9.6
[1,1,2,1]	6.08
[1,2,1,1]	5.19
[3,3,3]	4.3

Table 1: *This table shows a comparison of parameters by varying the number of blocks per residual layer, where each element represents the number of blocks per residual layer. It was observed that the addition of blocks to the end layers results in more parameters being added compared to the initial layers.*

In Table 1, we can see that using [2,1,1,1] blocks per residual layer results in a model with fewer than 5 million parameters. Additionally, by reducing the number of channels to 36, 64, 128, and 256 instead of 64, 128, 256, and 512 channels, while keeping the original layer structure [2,2,2,2], we obtained a test accuracy of nearly 94% after 200 epochs, with a training accuracy exceeding 99%. However, this high training accuracy indicates that the model may be overfitting the data.

We also experimented with removing the last 512 blocks from the ResNet-18 architecture and using a [3,3,3] block instead. This modification resulted in a model with approximately 4.3 million parameters, and after 200 epochs, it showed slightly better accuracy and loss results than the previous approach.

Overall, we found that the optimal layer structures were [2,1,1,1] and [3,3,3], as they had less than 5 million parameters and achieved a high test accuracy of over 94%.

But,due to time constrains,we are training the model for 80 epochs and we get the following accuracy as 86.24% and 86.12% for [2,1,1,1] and [3,3,3].

Data Augmentation

To enhance the performance and results of the modified model, we utilized data augmentation techniques on the training set. This involved generating new and diverse instances for the training dataset. To achieve this, we utilized functions like RandomRotation(), RandomCrop() and RandomHorizontalFlip() from the torchvision library to flip and crop the tensor images in the dataset. Finally, we used the Normalize() function to normalize the tensor image with the provided mean and standard deviation at the last stage of data augmentation.

Optimizers

The choice of optimizer can significantly impact the learning process. In our experiments, we tested various optimizers, including SGD, Adam, Adagrad, and Adadelta. We observed that the changes in loss and accuracy varied depending on the optimizer used. Among all the optimizers tested, SGD consistently resulted in the highest accuracy for the modified ResNet model, even after 200 epochs. Therefore, we have chosen SGD as the default optimizer for any future experiments.

Epochs, Learning rate, Weight decay & more

We conducted further experiments on various hyperparameters such as Epochs, Learning rate, Weight decay, Tmax, and more. We tested Epochs ranging from 100 to 300 and found that 80 epochs was the optimal value where the train and test accuracies plateaued. For the Learning rate, we tried values from 0.001 to 0.1 and found 0.1 to be stable and provided the best accuracy.

Regarding Weight decay and Tmax, we followed the same values as the original ResNet-18 model by Kaiming He et al. which was 5e-4 and 200, respectively. Our batch sizes for training and testing were 128 and 100, respectively. We also experimented with filters, strides, and max pool instead of average pooling, but none of them showed significant improvements, so we used the default values from the original ResNet-18 model.

Initially, we trained our model on the CIFAR-10 dataset, and by using different architectures, we achieved just over 95% test accuracy. However, some of the approaches overfit the model. To overcome this issue, we introduced a validation dataset, which reduced the accuracy for the test dataset slightly but maintained the same pattern of train accuracies. In general, the test accuracies for several approaches were near 86

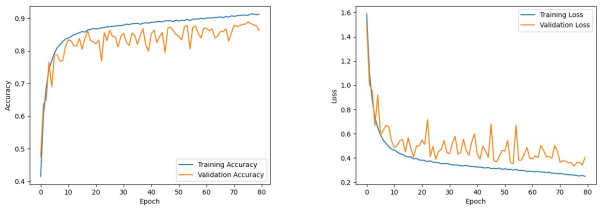
Regularization

In order to prevent over-fitting, we decided to apply Dropout regularization techniques to our model. However, we wanted to take it a step further by applying dropout specifically to the convolution and/or average pool layers in each channel size category. This approach is not commonly used as it offers limited benefits, but our goal was to maximize test accuracy as much as possible. By using dropout on convolution or average/max pool layers, we are not disabling a few weights but rather setting specific entries in the output to zero, which offers limited benefits, as stated by Jacob Reinhold in 2019. To prevent our model from overtraining, we used early stopping with a patience of five epochs. This approach not only reduces overfitting on the dataset but also improves the generalization of the trained model.

Results

We conducted experiments with different combinations of optimizers, parameters, and ResNet model architectures, within the given parameter constraints. While many approaches produced similar results, this section explains the best models we could develop. We observed that two main configurations exhibited competitive test accuracies:

- **Architecture 1**: With Conv layers [2,1,1,1] block configuration for 64, 128, 256, and 512 channels respectively.



(a) Arch-1: Accuracy Vs Epochs (b) Arch-1: Losses Vs Epochs

Figure 1: *Architecture-1*

- **Architecture 2**: With 512 channel block removed, using only [3,3,3] block configuration for 64, 128, 256 channels respectively, plus dropout for Conv layers and an extra hidden linear layer of 128 neurons.

N	Bi	Dropout	Acc(%)
4	[2,1,1,1]	-	86.24%
3	[3,3,3]	0.2 (Conv layers + hidden layer)	86.12%

Table 2: Parameter Comparison

Table 2: N : Residual Layer, Bi: Residual Blocks, Dr : Dropout Rate Acc : Test Accuracy

Architecture-1

The initial model design has a parameter count of less than 5 million (around 4.97 million) and results in a test accuracy of 94.17% after 200 training epochs. However, the training accuracy exceeds 99%, indicating that the model is overfitting. Incorporating additional regularization techniques, such as dropout,taking care of time constrains and reducing the epochs to 80 further reduces the test accuracy to approximately 86%. Please see Figures 1 and 2 for more details.

Architecture-2

The [3,3,3] model has approximately 4.3 million parameters and achieves a test accuracy of 94.33% after 200 epochs with 20% dropout applied to the Convolutional layers and a hidden layer with 128 units. This model architecture consists of two hidden dense layers with sizes 256, 128, and 10 (for the CIFAR-10 classes). The additional dense layer was included to better capture arbitrary boundaries with the ReLU activation function and to achieve a smoother classification of the classes. Dropout regularization was used to mitigate overfitting and align the train accuracy as closely as possible with the test accuracy. Further reducing the epoch to 80,we got accuracy of around 86%. Increasing the dropout from 20% to 40% slightly reduces the test accuracy, but does not result in a significant reduction in the training accuracy. Please see Figures 3 and 4 for more information.

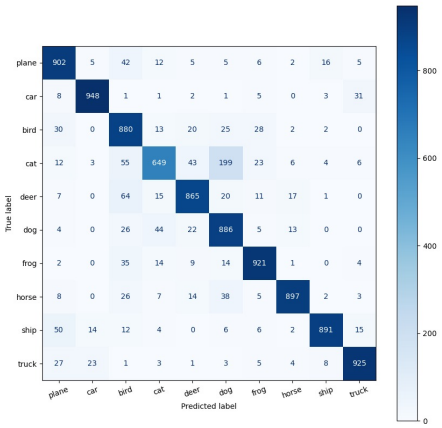
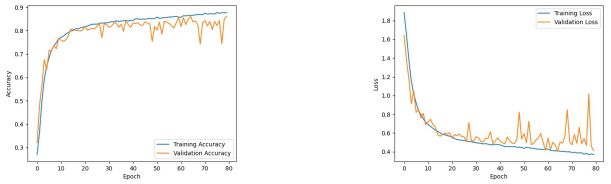


Figure 2: *Confusion Matrix for Arch-1: [2,1,1,1] ResNet without any dropout*



(a) Arch-2: Accuracy Vs Epochs (b) Arch-2: Losses Vs Epochs

Figure 3: *Architecture-2*

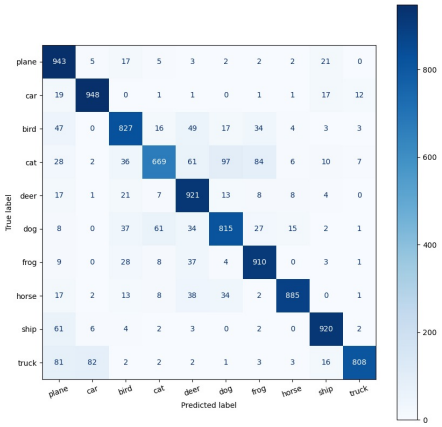


Figure 4: *Confusion Matrix for Arch-2: [3,3,3] ResNet with 0.2 dropout*

Conclusion

Throughout our experiments, our goal has been to keep the number of trainable parameters under 5 million, achieve a test accuracy of at least 86%, and prevent overfitting of the model to the training data. We have explored numerous combinations of residual layers, blocks, optimizers, and regularization techniques, including more subtle modifications such as changing convolution strides. Our reference architecture is based on Kaiming He's work, and our main challenge has been to implement dropout and its related effects, as even with the addition of dropout (with different probabilities), overfitting remained a problem.

One notable finding was that Architecture 2, with a [3,3,3] block configuration, a dropout rate of 20% for Conv layers and a hidden layer, achieved a validation accuracy of above 85% around 80 epochs, with training accuracy around 90%. Compared to Architecture 1, which lacked dropout, this model exhibited smoother losses and more stable train and validation accuracies. Further testing revealed a maximum test accuracy of 86%. In future research, we plan to experiment with higher dropout values and different hidden layer configurations to optimize the model further.

References

Great Learning. 2022. Introduction to Resnet or Residual Network. https://www.mygreatlearning.com/blog/resnet/. Accessed: 2022-11-12.

Jacob Reinhold. 2019. Dropout on convolutional layers is weird. https://towardsdatascience.com/dropout-on-convolutional-layers-is-weird-5c6ab14f19b2. Accessed: 2022-11-09.

Kaiming, H. 2016. Deep Residual Learning for Image Recognition. IEEE conference on computer vision and pattern recognition, 770–778.

Kaiming He. 2018. Deep Residual Learning for Image Recognition. https://github.com/kuangliu/pytorch-cifar. Accessed: 2022-11-01.

Nikolas Adaloglou. 2020. Intuitive Explanation of Skip Connections in Deep Learning. https://theaisummer.com/skip-connections. Accessed: 2022-11-11.

Pablo Ruiz. 2018. Understanding and visualizing ResNets. https://towardsdatascience.com/understanding- and-visualizing-resnets-442284831be8. Accessed: 2022- 11-12.

Qizhe, Z. 2020. Self-training with Noisy Student improves ImageNet classification. Xie, Qizhe, Minh-Thang Luong, Eduard Hovy, and Quoc V. Le. "Self-training with noisy student improves imagenet classification." In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 10687–10698.