Classification of images in CIFAR-10 using ResNet architechture

Written by Shrey Jasuja, Prasanna Sai Puvvada, Rahul Raj

Codebase: Github Colab
Tandon School of Engineering
New York University

Abstract

Residual Networks (ResNets) were developed to address the vanishing gradient problem by introducing residual blocks, and subsequent research by Hao Li et al. (2018) has shown that skip connections make the loss landscape smoother. ResNets enable training of very deep networks, which was not possible before. In this study, we explore custom ResNet architectures for classifying the CIFAR-10 image dataset. CIFAR-10 is a small dataset with 10 classes, making it a suitable testbed for evaluating design choices. We performed approximately 15 experiments, varying the number of layers, filters, dropout rates, and learning rate schedulers. We present the results of five experiments with contrasting settings, including a 56-layer ResNet trained with a constant learning rate of 0.01, an 80-layer ResNet (which gave the best results) with a dropout rate of 0.5, and a OneCycle learning rate scheduler, achieving an accuracy of 95.16%. Our results demonstrate the effectiveness of ResNet architectures and the importance of optimizing learning rate schedules and dropout rates in deep neural networks.

Introduction

Deep neural networks have achieved remarkable success in computer vision tasks, especially in image classification. However, as the depth of the network increases, the accuracy often saturates or even degrades rapidly, known as the degradation problem. To address this issue, the authors of ResNet introduced residual learning, where the network learns the residual mapping instead of the underlying mapping. This approach allows for the training of very deep networks and has been successful in various computer vision tasks, particularly in image classification.

To implement this idea, ResNet introduces shortcut connections that skip one or more layers and directly connect the input to the output. The output of these shortcut connections is added to the output of the stacked layers, and the sum is passed through a nonlinearity. These shortcut connections enable the network to learn the residual mapping more efficiently and effectively, allowing the network to be deeper without suffering from the degradation problem.

Let us take $H(x)$ as an underlying mapping that is fitted by few stacked layers which does not mean complete net necessarily and let nonlinear layers fit another mapping of $F(x) = H(x) - x$ where x is inputs to the initial layers. Recasting the original mapping to $F(x) + x$. The original mapping is simpler to optimize than the original, unreferenced mapping. In the extreme, fitting an identity mapping by a stack of nonlinear layers would be more difficult than pushing the residual to zero if an identity mapping were best. Feedforward neural networks with "shortcut connections" can be used to implement the formulation of $F(x) + x$ (Figure. 1). those connections that skip one or more.

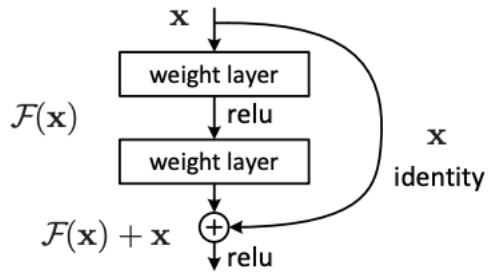

Figure 1: Residual learning: a building block

$F(x)$ is a block that implements
$conv \rightarrow BN \rightarrow relu \rightarrow conv \rightarrow BN$ here, "BN" stands for batch normalization, conv is convolution. Chaining such blocks serially gives a deep ResNet.

In this instance, Shortcut connections perform identity mapping, adding results to stacked layers without increasing complexity or introducing new parameters (Figure 1). The network is trained end-to-end with SGD and backpropagation, easily built using open-source frameworks.

In this paper, we investigate the performance of ResNet on an image classification task and analyze the effect of different network configurations, such as the number of layers and the width of the network. We also study the impact of various hyperparameters, such as the learning rate, number of layers, and number of filters. Our exper-

iments demonstrate that ResNet achieves state-of-the-art performance on the image classification task and is highly robust to different hyperparameter settings.

Methodology

1. Data Augmentation

Since the CIFAR-10 dataset has low-resolution images, we used simple augmentations as mentioned in the original paper. During training, we added 4 pixels of padding on each side, and randomly sampled a 32×32 crop from the padded image or its horizontal flip. During testing, we only evaluated the single view of the original 32×32 image.

2. Network Architecture

2.1 Types of Blocks

The original paper introduces two types of residual blocks: basic and bottleneck. Basic residual blocks are used in the simple ResNet architecture, while bottleneck blocks are more suited for networks with a very large number of filters, where they can lower the required number of parameters and memory. For our experiments, we used only the basic residual block.

2.2 Number of Filters

The number of filters in the convolutional layers can improve the network's ability to extract relevant features and patterns from the input data. Since CIFAR-10 images have low resolution, we experimented with starting from 16 filters and gradually increasing them to 64 filters towards the last layers. We also tried using more filters and scaled up to 128 filters in the last layer, making the network wider.

2.3 Number of Layers

We experimented with different numbers of layers, including 56-layer, 128-layer, and 80-layer networks. We found that deeper networks tend to have better accuracy, thanks to the skip-connections introduced in the ResNet architecture, which enabled us to train deep networks without much difficulty.

2.4 Layers

Layer Structure The ResNet model used for the classification of CIFAR-10 is illustrated in Figure 2.

We experimented with several architectures based on the ResNet model, including a 128-layer network. Here, we describe the architecture of the 128-layer network.

Convolution 1, Layer 1: In ResNet, the first layer is a 3x3 convolution with batch normalization and rectified linear unit (ReLU) activation. The stride for this layer is 1, and padding is 1 to match the output size with the input size. The number of input channels is 3, and the number of output channels is 16.

The output volume of Convolution 1/Layer 1 is 16x32x32, as shown in Figure 3.

Layer 2: We use a stack of 21 layers, which determines the size of our ResNet. The feature map size is 32 with 2 convolutions, and the number of filters is 16 for this layer.

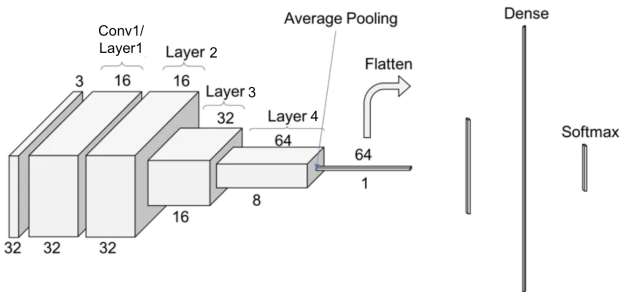

Figure 2: ResNet architecture for CIFAR-10 classification

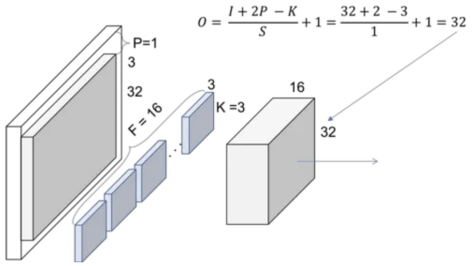

Figure 3: Convolution 1/Layer 1

The down-sampling of volumes in this ResNet model is achieved by increasing the stride to 2, but for Layer 2, we use a stride of 1 and a filter size of 16. No projections are used for skip connections, so the input channels are the same as the output channels, and the output size is matched before addition.

The output volume of Layer 2 is 32x32x16, as shown in Figure 4.

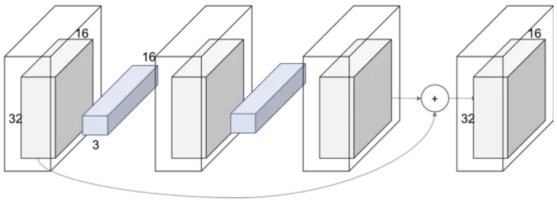

Figure 4: Layer 2

Layer 3 and Layer 4: Layers 3 and 4 behave similarly to Layer 2, but with a stride value of 2. This results in a reduction of the output volume by half, which means that the skipped connections require additional steps to adjust the sizes before summation.

For Layer 3, we use a feature map of size 16 with a filter size of 32, resulting in an output size of 16x16x32, as shown in Figure 5 and Figure 6.

For Layer 4, we use feature map of size 8 with a filter size of 64, resulting in an output size of 8x8x64, as shown in Figure 7.

Fully Connected Layer: After all previous layers, a dropout with a probability of 0.5 is added to prevent over-

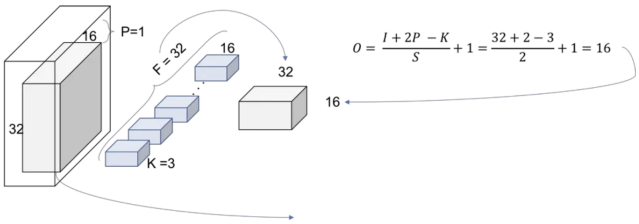

Figure 5: Layer 3, Block 1, Convolution

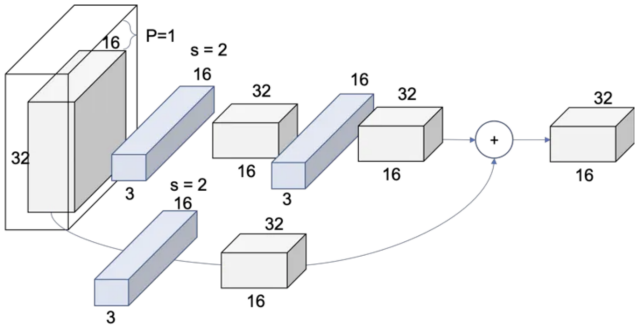

Figure 6: Layer 3

fitting, followed by an average pooling layer. Finally, a fully connected layer with 10 output classes is added.

2.5 Activation Function

ReLU is a simple non-linear activation function that has become a go-to choice for many deep learning applications. It has several advantages over other activation functions, such as its computational efficiency and its ability to avoid the problem of vanishing gradients. ReLU returns the input if it is positive, and zero otherwise. The output of ReLU is then fed into the next layer of the neural network.

3. Loss Function

To measure the goodness of our model, we used the Cross Entropy loss function, which is commonly used for classification tasks. Cross Entropy loss is a measure of the difference between the predicted probabilities and the true labels. It penalizes the model for making incorrect predictions and incentivizes it to assign high probabilities to the correct classes.

4. Optimization

4.1 Optimizer

To optimize our model, we used the Stochastic Gradient Descent (SGD) optimizer, which is well-suited for image classification tasks. We also used a momentum of 0.9 and a weight decay of 1e-4 as suggested in the literature.

4.2 Regularization

Overfitting is a common problem in deep learning, where the model learns to fit the training data too well and is unable to generalize to new data. To prevent overfitting, we

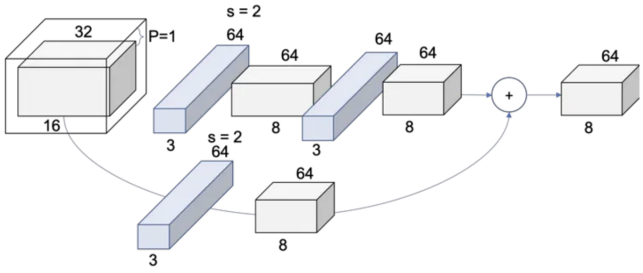

Figure 7: Architecture of Layer 4

used dropout regularization, a technique commonly used in deep learning. Dropout regularization helps to prevent overfitting by forcing the network to learn more robust and diverse features. It also reduces the number of parameters and computational resources required by the network, making it more efficient and faster to train. We experimented with both 2D-dropout and 1D-dropout. 2D- dropout was used between CNN layers, where it randomly sets a fraction of the channels to zero for each input during training. However, we found that 2D-dropout made it difficult to fit the model. On the other hand, 1D-dropout between the pooling layer and fully-connected layer helped to regularize better.

4.3 Learning Rate Scheduler

The choice of learning rate and learning rate scheduler can greatly affect the accuracy and convergence of a deep learning model. For our shorter networks, we chose MultiStepLR from Pytorch, which allowed the learning rate to decay by gamma (0.1 in our case) at given epoch milestones. For larger networks, we used triangular LR schedulers such as OneCycleLR, which first increases the learning rate, then decreases it after reaching the maximum. In all cases, the maximum learning rate was set to 0.1. We found that using a fixed learning rate was not effective in converging our models.

Experimental Settings and Results:

In this study, we conducted approximately 15 experiments to optimize a simple ResNet architecture which are mentioned in the Methodology section. These experiments involved changes in the number of layers, number of filters, dropout rates, and learning rate schedules. We will briefly discuss five experiments that highlight contrasting settings and report the resulting accuracy.

1: We trained a 56-layer ResNet with a constant learning rate of 0.01 and achieved an accuracy of 91.12% after training for 130 epochs.

2: We added a MultiStep learning rate scheduler with gamma=0.1 and max initial LR=0.1 to the 56-layer ResNet. However, the accuracy dropped to 89.51% due to suboptimal milestone values. We optimized the milestone values to 60, 85, and 110 epochs and allowed the model to struggle at higher LR values. This led to better convergence with an

accuracy of 92.84%.

3: We applied 2D dropout with p=0.1 to 0.2 on CNN layers themselves, in addition to 1D dropout with p=0.5 before linear layers. However, these models performed poorly with an accuracy of only 90.72%.

4: We increased the number of layers to 128 and found that a higher initial learning rate did not yield better results. Thus, we used the OneCycle learning rate schedule, which gradually dropped the LR for subsequent epochs after starting from 0.01 LR for 40% of epochs. With 130 total epochs, we achieved an accuracy of 93.75%.

5: We increased the number of channels to start with to 32 and set the last layer to have 128 channels. We used the same optimization settings as Experiment 4 and achieved an accuracy of 95.16%.

Overall, our experiments demonstrate the importance of optimizing the learning rate schedule and choosing appropriate dropout rates for different layers of the network. Moreover, increasing the depth and width of the network can lead to higher accuracy, our 80-layer network being wider performed better than 128-layer network, but at the cost of a higher number of parameters leading to approximately double memory.

Conclusion

Based on the results of our experiments, we can conclude that optimizing the learning rate schedule and dropout rates for different layers of the network is crucial to achieving high accuracy in ResNet architectures. We observed that increasing the depth and width of the network can lead to higher accuracy, but this requires careful tuning of the learning rate schedule to prevent overfitting.

In Experiment 1, we observed that a constant learning rate of 0.01 is effective in training a 56-layer ResNet. However, in Experiment 2, we found that adding a MultiStep learning rate scheduler with suboptimal milestone values can lead to reduced accuracy. By optimizing the milestone values in Experiment 2, we achieved higher accuracy.

Experiment 3 showed that applying 2D dropout on CNN layers themselves did not improve accuracy and in fact, led to lower accuracy compared to 1D dropout before linear layers. We observed that increasing the number of layers in Experiment 4 required the use of a OneCycle learning rate schedule to achieve better convergence and accuracy.

Finally, in Experiment 5, we increased the number of channels at the start and the last layer of the network, Making the network wider, which led to the highest accuracy of 95.16%. Our experiments demonstrate that different settings and combinations of hyperparameters can significantly affect the accuracy of ResNet architectures, and careful experimentation and optimization are necessary to achieve the best results.

Layers	Filters in first layer	Parameters	Model Size (in MB)	Highest Accuracy %
56	16	855,770	17.62	92.84 %
128	16	2,022,362	40.45	93.75 %
80	32	4,964,266	59.89	95.16 %

Table 1: Results with different Layers and Filters

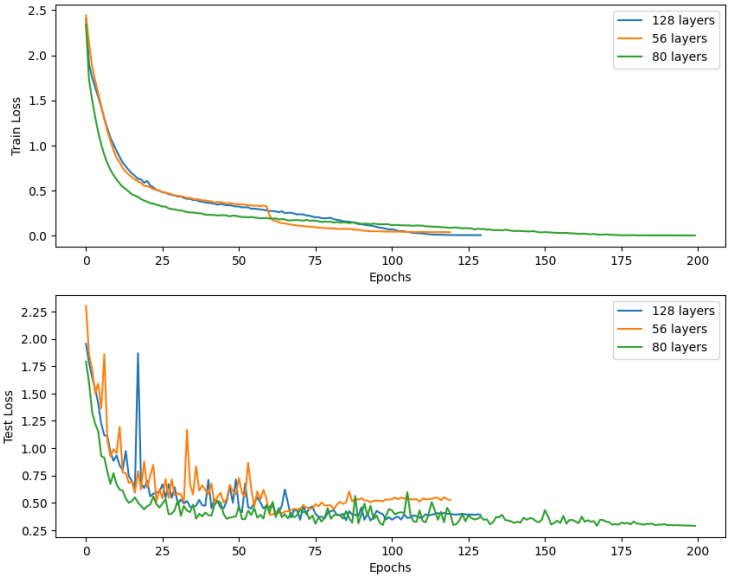

Figure 8: Different models comparison

References

He, K., Zhang, X., Ren, S., Sun, J. (2015). *Deep Residual Learning for Image Recognition.* ArXiv. /abs/1512.03385

Smith, L. N., Topin, N. (2017). *Super-Convergence: Very Fast Training of Neural Networks Using Large Learning Rates*. ArXiv. /abs/1708.07120

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R. (2014). *Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research, 15(1), 1929-1958.*

Pablo Ruiz (2018) *ResNets for CIFAR-10*

Nash, R. (2015). *An Introduction to Convolutional Neural Networks.* ArXiv. /abs/1511.08458

Alex Krizhevsky (2017) *CIFAR-10 datasets*