Architect constrained ResNet for CIFAR-10 Classification

Arjun Naga Siddappa, Atsushi Shimizu, Karthik Udhayakumar

New York University

Abstract

We propose creative image classification models with a ResNet block with no more than 5 million parameters. Our model trained on CIFAR-10 with relatively deep network attains 93.2% of accuracy on test set, while another model with ensembling reaches 94.47% of accuracy. We also give an extensive discussion not only on what works but also on what doesn't work. Through this trial and error, our learning experience was maximized, and we obtained a great intuition about how the ResNet behaves.

Introduction

In this project, we are tasked to perform classification on the CIFAR10 dataset using ResNet architecture. It starts with a convolutional layer, then a set of residual layers made out of Residual blocks, followed by average pooling layer and it ends with the fully connected layer with softmax function. The building blocks of the ResNet model [1] consists of 2 convolutional layers back to back together with a 'skip' connection from the input to output. This skip connection is the novelty of the model as it helps the layers to learn identity function more easily and solves the vanishing gradient problem. This helps us train deeper networks without losing on performance. As discussed in our lectures about the 3-step recipe, ResNet is the model with which we aim to classify, Cross Entropy Loss is the loss function and Optimizer can be Stochastic Gradient Descent (SGD). The aim of our project is to achieve highest accuracy we can under the constraint of 5 million parameters for the ResNet model.

Methodology

0. Baseline Architecture

We begin with the baseline architecture of the ResNet as in [2]. It has the 11.17 million parameters with 4 residual layers of channel sizes of 64,128,256,512 respectively. We used Cosine Annealing for learning rate scheduler with an initial learning rate of 0.1 and SGD and we trained the model for 100 epochs which yielded the test accuracy of 86.8 . Our goal is to make architectural changes to ResNet to bring the number of parameters to less than 5 million but also experiment with the choice of learning rate, optimizer to achieve or even improve upon the test accuracy we got for the baseline architecture.

1. Channel Size

Since there is a limit on how many parameters we want in our model and given the base model with 11.17 million parameters, we perform a number of changes to this base model to bring down the no. of parameters. One of the ways is the change the channel sizes of the convolutional layers in the residual blocks. The channel sizes in the four residual blocks initially were 64,128,256,512 respectively. We change it to 16, 64, 128, 256 respectively. This results in 90.3% accuracy.

2. Optimizer

The base line model was trained with Stochastic Gradient Descent as the optimizer. We wanted to experiment with Adam Optimizer hoping to see an increase in the performance. With the decreased channel size model from 2) we trained the model with Adam Optimizer. This gave an accuracy of 80% as well

3. Learning Rate Management

In the original code in the GitHub, the authors' used CosineAnnealingLR for the learning rate scheduler. The initial learning rate was set to 0.1, and this is decreased to 0.0 at 200 epochs, following the cosine curve. On the other hand, the paper mentioned another approach of dividing the learning rate by 10 at 32,000 and 48,000 iterations. To mimic this setting with our limited computational resource, we use another scheduler, called ReduceLROnPlateau, to keep track of the test accuracy, and divide the learning rate by 10 if no improvement in the fixed number of epochs. The initial learning rate is set to 0.1 in both approaches and compare the performances by training 100 epochs. So, $T_{\max}$ is set to 100 in the CosineAnnealingLR.

4. Deeper Network

One of the advantages of ResNet is its ability to train large number (even thousands) of layers without increasing the training error percentage. Hence we wanted to explore how a deeper model would perform. Based on the results from changes in Channel Sizes as well, we knew that we could decrease the channel size for each residual layer. We ended up with ResNet with 8 residual layers of channel sizes 16,32,64,96,128,160,192,220. The channel sizes are in increasing order of 32 considering our constraint of 5 million parameters. Since the model is deep and our input is just 32x32, we also experimented with the stride size of convolutional layer. We ended up with two models, first with stride size as the same from the baseline architecture which is 2, while the second model has a stride size of 2 for every alternating residual layer only. This helped in persisting the input image's features in the deeper layers of the network. We trained both models using ReduceLROnPlateau as the scheduler and 100 epochs.

5. Shallow Network

One of common patterns in neural networks is, as the model gets deeper the number trainable parameters increases. Although this helps the model learn and generalize better, this comes at a cost of training time, and performance. Hence to reduce the number of parameters and bring it under 5 million, we removed the last residual layer of channel size 512, which yielded a ResNet with 3 residual layers of channel sizes 64,128,256. This new architecture has only 2,777,674 parameters. To compensate for the loss of trainable parameters, we experimented with starting learing rates of CosineAnnealing, giving values 0.1 and 0.01. Additonally, Based on the results from Learning Rate Management, we decided to combine this model with ReduceLRonPlateau scheduler to update the learning rate if there is no improvement on the specified number of epochs. Hence, We expect this approach to converge at an optimal result quickly and trained this model for 100 epochs as well.

6. Skip Connection Manipulation

6.1 Kernel Sizing ResNet is different from other algorithms in that it has skip connections. These skip connections are basically convolutional layers. Thus, they lend to hyperparameter tuning. The kernel size of these skip connections in the baseline model is 1. We wanted to experiment with this size. Making the kernel size 3 with a padding of 1 would give the same sized output as the previous skip connection. The accuracy didn't improve much with a value of 91.88.

6.2 Identity Mapping Another method proposed in the paper is a simple identity mapping with zero-padding for increasing dimensions in the skip connection where the number of channels changes before and after the regular 2-layer block computation. Unlike the 1×1 filters require us to budget parameters, this identity mapping is done with no additional parameters. As the paper doesn't mention how to reduce the image size, we implement the average pooling, which is also a parameter-free operation, to compress the image size.

7. Dense Layers

Usually when convolutional layers are used in a neural network they are followed by fully connected linear layers. In ResNet we see one linear output layer at the end. We wanted to experiment by adding more linear layers. We added two linear layers of size 128, 64 before the final layer. The total number of parameters is 3,151,786 and the accuracy is 90.74 at 100 epcohs.

Model	# Params	Test Accuracy (%)
Deeper net	4,997,386	93.2
Shallow net	2,777,674	94.55
5-Ensemble with Averaging	4,937,010	94.47

Table 1: Top 3 Models

8. Ensembling

One of our observations in the early stages of the project was that smaller networks, such as having only 1 million or less parameters, are still very powerful, often achieving over 90% accuracy on the test set. So, in addition to building a single model with 5 million parameters, we also train multiple small models and aggregate the outputs to obtain a final prediction. Under the constraint of no more than 5 million parameters, it is a natural assumption that there is a trade-off between the number of small models and the powerfulness of each individual model. Trying to find an optimal balance of them, we examine 5-model ensembling with each less than 1 million parameters, and 10-model ensembling with each less than 500,000 parameters. As an aggregation method, we see two approaches; the majority vote and averaging out the softmax outputs.

Main Result

The top 3 architectures in terms of test accuracy are summarized in Table 1.

Discussion

In this section, we shall see the results of the each experiment we performed for every methodology explained in the earlier section. We also try to reason behind the results and the significance of the methodology.

1. Channel Size

We see a drastic increase in the accuracy from 80% in the baseline model to 90%. Since our dataset has fewer classes a small model would suffice and a smaller model is faster to train and is less prone to overfit than a bigger model. That is why we at 100 epochs we see the smaller model having an accuracy of 91.54% and the bigger model giving us 80%.

2. Optimizer

We changed the optimizer from SGD to Adam. We expected Adam optimizer to give us better performance but not sure why but the SGD seems to give a better and faster training with a higher test accuracy sooner.

3. Learning Rate Management

We conduct an experiment where we compare the performance of the CosineAnnealingLR and the ReduceLROnPlateau. We see the final test accuracy is very much the same for both methods. However with the CosineAnnealingLR, the number of training epochs becomes the hyperparameter and it would be difficult to tell what value is good for each model without searching it. Considering we train several models with different architectures, the choice of ReduceLROnPlateau is more reasonable for us because it manipulates the learning rate by the model complexity, and we can easily continue training if the initial epochs are not sufficient.

4. Deeper Network

The initial model with an constant stride of 2 resulted in a test accuracy of 91.21. Although, this seems to be a reasonable accuracy, upon inspecting the model, we noticed that last 3 resulting layers has an input shape of 1x1. Unlike the ImageNet dataset used in the original ResNet paper, the CIFAR10 dataset used here has images of shape 32x32. This results in a shape of input being halved by using the stride of size 2 essentially rendering the last three residual layers train on input of shapes 1x1. Hence to persist information deeper into the network. Stride size 2 is applied only to alternating residual layers. As expected, this model performed better as it was able to train the deeper network on a larger input shape when compared to the earlier model. After 100 epochs the newer model yielded an test accuracy of 93.2. From the result we can infer that ResNet exploited this thinner and deeper network, persisting the convolutions deeper into the layers.

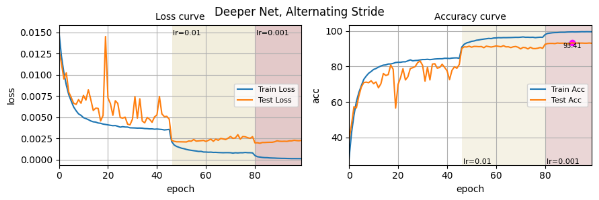

Figure 1: Deeper Net, with Alternating Stride between residual layers

5. Shallow Network

Of the three approaches we experimented, CosineAnnealing with starting learning rate of 0.1 yielded a test accuracy of 87.3 and 0.01 yield 90.6 . But the best accuracy resulted from using ReduceLROnPlateau which yielded a test accuracy of 94.55 after just 100 epochs. Training more than 100 epochs lacked any motivation as the training error was almost close to zero around 100 epochs. We believe, this model performed well because of the input dataset CIFAR which has images of shape 32x32, which the model was able to learn better and generalize under 5 million parameters when compared to the original paper, where ResNet model was tasked to classify images of size 224x224 from the ImageNet dataset.

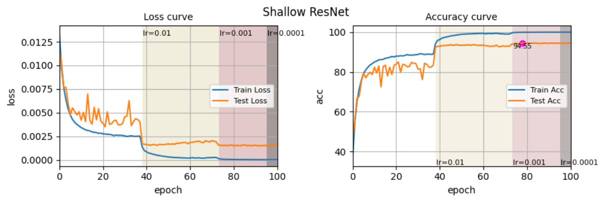

Figure 2: Shallow ResNet

6. Skip Connection Manipulation

6.1 Kernel Sizing We expected a higher kernel size of the convolutional layers in the skip connection to give a better performance. We changed the kernel size from 1 to 3 with padding of 1. This did not give us as much of a boost in our model as we were expecting. We see an increase from 91.54 to 91.88. Maybe the kernel size of the convolutional layers in skip connection do not influence much.

6.2 Identity Mapping In our experiment, two models with and without 1×1 filters in skip connection with roughly 2.7 million parameters are constructed. They are trained 100 epochs in the same hyperparameter settings. The result are 88.31% test accuracy in identity mapping model, and 88.59% in the one with filters. This result, identity mapping is slightly worse but the difference is small, is consistent with what the paper argues. If the identity mapping reduces the number of parameters significantly, we could have used it for other parts of the network. However, the parameters are saved only 1.57% by the identity mapping.

7. Dense Layers

Our intuition behind this experiment was that the residual blocks would serve as feature-extractors and the dense layers would do the image classification. The new model although around the same range performs not as good as the model without the dense layers. We see a decrease in accuracy from 91.54% to 90.74% at 100 epochs. It maybe the case that much of the learning is done in the residual blocks and the dense layers are adding unnecessary complexity.

8. Ensembling

5-model ensembling with averaging out the softmax outputs shows the strongest test accuracy of 94.47%. The loss and accuracy curve is shown in the figure 3. Considering the accuracy of each small predictor is at most 93.47%, mixing out various opinions from different predictors pushes the test accuracy by as much as 1.0%. The majority voting results in 94.41%. In 10-model ensembling, the test accuracy is 94.23% and 94.06%, respectively. The best accuracy of each small predictor is 92.30%.
During training multiple small models, we also obtained several interesting observations. Firstly, we need the regularization because the training error gets very close to zero oftentimes. However, incurring the ℓ_2 regularization through weight decay parameter doesn't bring much difference. Instead, stronger modification of training data through augmentation, such as allowing rotations up to 30 degrees, works as a regularization. Second, decreasing the learning

rate more carefully, namely dividing not by $1/10$ but $1/\sqrt{10}$, doesn't show apparent difference.

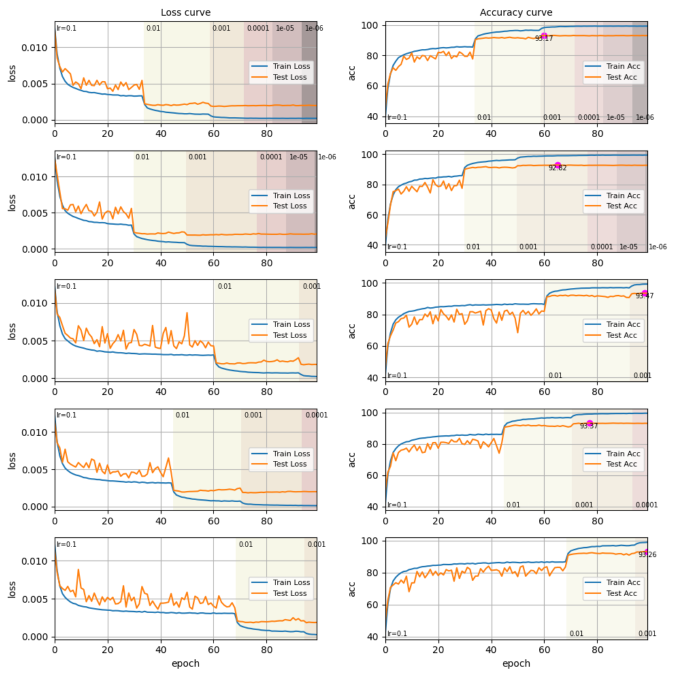

Figure 3: 5-Ensemble Training

Conclusion

Out of the various experiments and approaches we had, we had two models that gave us the highest accuracy: the Shallow Network model (94.55%) and the Ensemble Model (94.47%). The one fact that we observed throughout different experiments is that smaller models performed better. This we believe is due to the small no. of classes and the small dataset. We obtained this high accuracy with limiting the number of parameters to 5 million.

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. "Deep Residual Learning for Image Recognition", Microsoft Research, 2015

[2] kuangliu. https://github.com/kuangliu/pytorch-cifar, Feb 2021

[3] D. Karani, "Experiments on hyperparameter tuning in deep learning-rules to follow," Medium, Nov 2020.

[4] Kmldas, "Cifar10 resnet: 90 plus % accuracy;less than 5 min," Jan 2021.