ResnetRS : Experimenting with training, scaling strategies and architectural changes

Apurva Purushotama
Department of Computer Science
Texas A&M University
College Station TX - 77843
apurvap011@tamu.edu

Abstract

ResNet was introduced to improve the performance of the model by increasing the number of layers. This project uses ResNetRS network which employs various training and regularization methods to improve the performance instead of making major architectural changes. This project implements ResNetRS and the suggested methods to handle overfitting and making the model more robust. A ResnetRS model was trained for 200 epochs on CIFAR 10 dataset with Autoaugmentation, CosineLR Decay, EMA and Label Smoothing. Some improvements were made over and on top of the existing strategies in the paper such as increasing the stem width, modifying the stride size, using CosineLR with warm restarts. With these improvements, the model was able to obtain an accuracy of **91.19%** on the test data sets.

1 Introduction

The neural network architectures such as VGG, AlexNet focus on increasing the number of layers to improve the performance of the model for image recognition and image classification. However, increase in depth can lead to several problems such as vanishing gradients. In order to alleviate this problem, ResNet includes residual blocks that contain 2 convolutional blocks. The skip connections allow the model to learn the model better. While most of the papers propose architectural improvements to increase the performance, Resnet-RS model proves that minor architectural changes can enhance the performance if proper training and regularization strategies are employed.

The training and regularization strategies, the architectural changes suggested in the paper **[1]** are summarized in Section 2 and Section 3. I have implemented most of changes according to the paper such as using SE block, Exponential Moving Average(EMA),Label Smoothing, Cosine LR Decay. However, there are a few different strategies that I experimented with while implementing the paper and are listed in Section 4.

Preprint. Under review.

2 Summarizing the ResnetRS Architectural changes

The Resnet architecture is being modified in the below ways in order to improve the performance. My understanding of the ResnetRS architecture are mentioned below.

Block Group	Output Size	Convolution Layout
stem	112x112	[3x3, 64, s2 3x3, 64 3x3, 64] x1
c2	56x56	[1x1, 64 3x3, 64 1x1, 256] x3
c3	28x28	[1x1, 128 3x3, 128 1x1, 512] x4
c4	14x14	[1x1, 256 3x3, 256 1x1, 1024] x23
c5	7x7	[1x1, 512 3x3, 512 1x1, 2048] x3
	1x1	[Avg Pool Dropout 1000-d FC] x1

Figure 1: Updated architecture for Resnet101 and input image size 224x224

2.1 Resnet-D

The difference between original Resnet and Resnet-D are as follows:

- There are three 3x3 conv blocks in the Stem Block instead of a single 7x7 block.
- For the downsampling blocks, the strides are changed for conv blocks of the residual path.
- Average pooling(stride=2, filter = 2x2) and 1x1 conv blocks are used in skip connection of downsampling blocks.
- The max pooling layer of Resnet is replaced with downsampling block before the first 3x3 conv of the bottleneck block

2.2 Squeeze and Excitation

The paper **[2]** implements the concept of squeeze and excitation. Since convolution works on principle of spatial locality and stationarity, the squeeze block here helps in extracting information slightly outside the receptive field as well. I have used Global average Pooling concept as mentioned in the paper. The excitation block uses the output of the squeeze process to map dependencies of multiple channels using Sigmoid function. These two operations - squeeze and excitation should help in improving the model accuracy.

3 Summarizing the Training and Regularization strategies

Figure 2[1] represents the list of changes and their corresponding improvements in accuracy. The Yellow ones represent the architectural changes(mentioned in Section 2), Green ones represent the different regularization methods and Purple ones represent the training methods.

Improvements	Top-1	Δ
ResNet-200	79.0	—
+ Cosine LR Decay	79.3	**+0.3**
+ Increase training epochs	78.8 †	-0.5
+ EMA of weights	79.1	**+0.3**
+ Label Smoothing	80.4	**+1.3**
+ Stochastic Depth	80.6	**+0.2**
+ RandAugment	81.0	**+0.4**
+ Dropout on FC	80.7 ‡	-0.3
+ Decrease weight decay	82.2	**+1.5**
+ Squeeze-and-Excitation	82.9	**+0.7**
+ ResNet-D	83.4	**+0.5**

Figure 2: Training and Regularization Strategies

Regularization strategies:

3.1 Data augmentation

[3]. By including a number of variations training samples in the training dataset via augmentation, the model can learn how to generalize better. The paper **[1]** suggests the use of RandAugment, an augmentation strategy that employs parameterization. It can be used for various model and dataset sizes. Out of the different transforms(Autocontrast, Rotate,Contrast,Brightness,TranslateX,TranslateY,rotate and so on) that have been defined, random list of policies are selected and operations are applied with a certain magniture.

3.2 Label smoothing

[4] For improved performance, the cross-entropy loss should be minimum. However, if the model assigns higher probability to the ground truth than the rest of them, there might be problems of overfitting on the training data. The model will not be able to adapt or generalize when it is fed with new data. In order to get a model that can adapt, label smoothing is proposed. Here, instead of using one-hot encoded labels, a smoothing parameter is added in order to prevent the model from becoming overconfident.

3.3 Exponential Moving Average

The main concept of EMA is to give more importance to the latest data. In terms of the model, I have to take EMA of model weights so that the weights of the latest epochs are given higher importance and the previous weights are also considered, but with lesser weightage.

3.4 Dropout

When the network has large number of parameters, the model can easily overfit. Dropout**[5]** is a regularization strategy where a fraction of neurons are dropped during the training phase in a random manner. This is said to improve the performance by a significant amount.

Training strategies:

3.5 Learning Rate Decay

Setting the correct learning rate while training and using optimization algorithms is an important factor. Deciding this hyper-parameter becomes difficult if we manually check the improvements since deeper networks take longer time to train. **"Cyclical Learning Rates for Training Neural Networks."** paper suggests to vary the learning rate over different iterations if a constant learning rate doesn't improve the model's accuracy. Cosine LR Decay is the suggested function based on which the learning rate should be varied.

4 Implementation and Observation

I implemented the strategies explained in Section 3 for the CIFAR-10 Dataset given.

4.1 Architecture

I have implemented the ResnetRS network. The StemBlock, bottleneck block, SE blocks were constructed and layers were made as per the architecture seen in Figure 1. ResnetRS50 model - a Bottleneck block with [3,4,6,3] layers was selected to be implemented. For SE Block, Reduction ratio=16 gave the best performing model

4.2 Improvement Strategies used

- **Label smoothing:** Label smoothing is used as the criterion in the model.train() instead of using cross entropy function. The model performance improved with label smoothing implementation.(**Check acc change now and add**). With this it alleviates the problems of calibration and model being overconfident. It was observed that the train accuracy improved where the test accuracy decreases sometimes as label smoothing penalizes the model that is overfitting.
- **Exponential Moving Average:** I implemented Class EMA to accumulate and update the model weights based on the exponential moving average computed. The EMA can be applied to model weights by wrapping the model within the EMA class written. The performance of the model with EMA was much better than the one without it.
- **Epochs:** Increasing the number of epochs allowed the model to train better. Since we are using CosLR strategy, the performance of the model(train and validation accuracy) decrease after the warm restarts. I kept a track of the train and validation loss for every 10 epochs to make sure that the model is not overfitting[Figure 3]**Add epoch vs loss graph here**
- **Dropout:** Dropout ratio of 0.25 was used.
- **Weight Decay:** According to the paper**[1]**, the default weight decay was changed from 1e-4 to 4e-5 since dropout is being used along with augmentation and label smoothing. This was done to avoid any over-regularization since too many regularization techniques are involved.

5 My Experimentation with training,regularization strategies and architectural changes

Apart from implementing the above stated methods, I experimented a few more strategies (training, regularization,and architectural changes) to check how the model's performance is being affected.

5.1 Architectural changes

(a) **Width of stem Block :** The stem_width of the Stem block was initially kept as 32. However, as the number of parameters increase, the model learns better. Hence, I tried increasing the stem width to check how the accuracy improves for a given number of epochs. With stem width = 128, the model performed way better than stem width = 32. The test accuracy improved from 89.17% to 90.28%

(b) **Stride change** Since the architecture in figure is for 224x224 image, stride=2 is used in the conv1 block of the Stemblock. However, since our image dimensions are 32x32, I tried experimenting with stride=1 for the Stemblock so that the output of the stemblock is not very small. Although this improved the train accuracy by a small amount, there was not any significant change in the test accuracy.

5.2 Experimenting with training and regularization strategies

(a) **Autoaugment** was used instead of RandAugment. Similar to Randaugment discussed in Section 3.1, autoaugment has a set of image processing functions like flip, crop, contrast, brightness etc in the form of subpolicies. Random policies are selected and operations are performed with some assigned magnitude. Although Autoaugment is computationally more expensive, it contributed well in improving the accuracy and generating more training samples. ResnetRS paper also mentions that the performance of the model would not significantly vary by using Autoaugment instead of Randaugment. Hence, I resorted to using Autoaugment itself.

(b) **Cutout:** This is another augmentation technique that is used as a regularization method. Cutout is used to mask the input image with squares. This helps the model learn the unmasked features of training data much better. It can classify images which have occlusion as well. I used 4 cutout squares, each of size 4x4 per 32x32 image. This improved the performance when compared to 1 cutout of size 16x16.

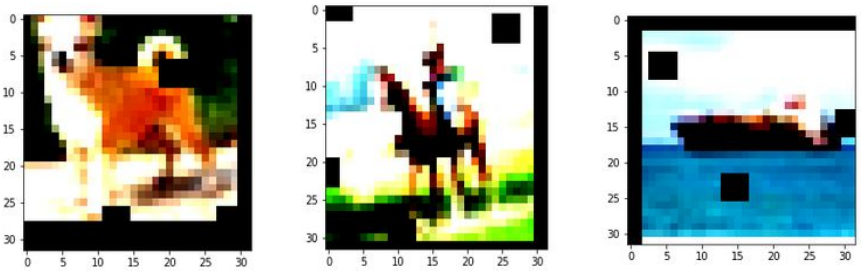

Figure 3: Augmented images with augmentation and cutouts

(c) **Optimizer:**SGD with Nesterov momentum was used.

(d) Paper **[1]** suggests to use **CosLR decay**. However, I experimented with other learning rate decay methods such as **MultistepLR** and **CosineAnnealingWarmRestarts**. Significant improvement in accuracy upon using CosineAnnealingWarmRestarts.The warm restart parameter (T0) which indicates the epoch number after which the LR is reset was set to [30,50,100] and 50 performed the best.

6 Results

Model	Learning Rate	Epochs	Stem Width	LR Decay	Test Accuracy
1	0.1	100	32	Multistep, gamma=0.2	87.23%
2	0.1	100	128	Multistep, gamma=0.2	89.07%
3	0.1	150	128	CosLRwithwarmrestarts	91.12%
4	0.1	200	128	CosLRwithwarmrestarts	**91.19%**

Figure 4: Observations with different changes

PyTorch Version : 1.7.0
ResnetRS network has been implemented using the training and scaling strategies described. With slight architectural changes, the model performance was improved. Also, using training and regularization methods to tune the model's performance is much easier when compared to changing the architecture. For the CIFAR-10 dataset, using the my strategies mentioned in Section 5, a test accuracy of **91.19%** was obtained on the test data that was available.

7 References

[1] Irwan Bello, William Fedus, Xianzhi Du, Ekin D. Cubuk, Aravind Srinivas, Tsung-Yi Lin , Jonathon Shlens, Barret Zoph **"Revisiting ResNets: Improved Training and Scaling Strategies"**
[2]Jie Hu, Li Shen, Samuel Albanie, Gang Sun, Enhua Wu **"Squeeze-and-Excitation Networks"**
[3]Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, Quoc V. Le **"RandAugment: Practical automated data augmentation with a reduced search space"**
[4]Rafael Müller , Simon Kornblith, Geoffrey Hinton**"When Does Label Smoothing Help?"**
[5]Geoffrey Hinton**"Dropout: A Simple Way to Prevent Neural Networks from Overfitting"**