Improved CIFAR-10 Image Classification with ResNet Architecture

Sang Tran

Abstract

This paper presents an attempt to replicate the results of the ResNet method on the CIFAR-10 dataset. The ResNet architecture, introduced in 2015, was a breakthrough in deep learning because it addressed the problem of degradation in very deep neural networks. Prior to ResNet, researchers had found that increasing the depth of a neural network beyond a certain point resulted in diminishing returns or even worse performance due to the vanishing gradient problem. Thus, it has achieved impressive results in image classification tasks, and my aim is to improve the accuracy by implementing similar architecture used in the original paper. The ResNet model is built using the PyTorch framework and trained on the CIFAR-10 dataset with different models. These experiments demonstrate that ResNet models achieve better results than other. This study highlights the importance of applying residual blocks to deep network, which results in verifying the efficacy of the state-of-the-art method and enable further research in the field.

1 Introduction

Image classification is a fundamental task in computer vision that has seen remarkable progress in recent years, with deep neural networks achieving state-of-the-art performance on various datasets. The CIFAR-10 dataset is a commonly used benchmark for image classification tasks, containing 50,000 training images and 10,000 testing images of 10 different classes. Among the deep neural network architectures that have achieved impressive results on CIFAR-10 is the ResNet method. By introducing residual blocks that allow for the creation of much deeper networks, ResNet achieved state-of-the-art performance on various image classification tasks, including the challenging ImageNet dataset. Since its introduction, ResNet has been widely adopted and improved upon in the deep learning community.

The goal of this paper is to show that applying residual blocks to simple architectures results in better accuracy than traditional architectures such as width scaling or depth scaling the network.

The importance of showing improved results of state-of-the-art methods cannot be overstated. My simplified ResNet models can serve as evidence why traditional methods are computationally expensive and do not show consistent results for future research in image classification tasks on the CIFAR-10 dataset. I believe that my efforts to applying the ResNet results on CIFAR-10 will contribute to the understanding and advancement of deep learning techniques for image classification tasks.

2 Method

2.1 Data

The experiments are conducted on the CIFAR-10 dataset, which consists of 50,000 training images and 10,000 test images, each with a size of 32x32 pixels and 3 channels. The standard train-test split is used across all models for evaluating the performance of each.

2.2 Models

There were six different models for the classification task on CIFAR-10. The models are as follows:

Model 1: A small model with 131,210 parameters consisting of 2 convolution layers, each followed by a ReLU activation and an average pooling layer. The output of the pooling layer is then flattened and fed to two fully connected layers, with the last layer being a softmax layer matching the number of classes in the dataset. Each convolution layer was argumented with kernel size of 3, stride 1 and padding 1.

Model 2: A deeper model with 3,425,284 parameters consisting of 5 convolution layers, each followed by a ReLU activation and there is an average pooling layer after the third and the fifth convolution layer. The output of the pooling layer is then flattened and fed to five fully connected layers with again, the last layer being a softmax layer matching the number of classes in the dataset. Each convolution layer was argumented with kernel size of 3, stride 1 and padding 1.

Model 3: A wider model with 16,064,620 parameters with the same architecture as model 1, although the numbers of channels and neurons are increased to a greater extent.

Model 4: A compound scaling model with 61,582,274 parameters consisting of 5 convolution layers, each followed by a ReLU activation and an average pooling layer. The output of the pooling layer is then flattened and fed to five fully connected layers with the aim of extending both the width and depth of the network. Each convolution layer was argumented with kernel size of 3, stride 1 and padding 1.

Model 5: A simplified ResNet model with 213,770 parameters consisting of three residual blocks, each containing two convolution layers, followed by a shortcut connection. The output goes through an average pooling layer and is then flattened and fed to two fully connected layers with the last layer being a softmax layer matching the number of classes in the dataset.

Model 6: Same as model 5 but applying batch normalization after each convolutional layer, with thus 214,442 parameters.

2.3 Training

The training process involves stochastic gradient descent (SGD) with a learning rate of 0.001 and momentum of 0.9 as the optimization algorithm for all of the models, which uses the cross-entropy loss function. Each model was trained for 10 epochs with a batch size of 4.

2.4 Hyper-parameters for models 5 and 6

Number of layers: 5.

Filter size: 3x3 for all the convolutional layers except 1x1 for the shortcut connection.

Stride size: The stride size for the first convolutional layer is 1, and the stride size for the average-pooling layers is 4.

Number of filters: The first convolutional layer has 16 filters, and each subsequent block doubles the number of filters.

Activation function: Linear Unit (ReLU) activation function.

Batch Normalization: Batch normalization is used for model 6 after each convolutional layer.

2.5 Evaluation

The performance of the models is evaluated on the test set using the classification accuracy metric.

3 Experiments

3.1 Results

The experimental results are summarized in Table 1. As can be seen, Model 4 achieved the highest accuracy of 90.23%, followed by Model 3 with an accuracy of 89.12%. Model 2 and Model 6 achieved similar accuracy of around 88%. Model 5 and Model 1 achieved the lowest accuracy among all models.

Model	Description	Number of Parameters	Avg. loss	Accuracy
1	Base	131,210	.607	66%
2	Depth Scaling	3,425,284	.367	63%
3	Width Scaling	16,064,620	.060	71%
4	Compound Scaling	61,582,274	.211	66%
5	Three ResBlocks	213,770	.528	76%
6	Three ResBlocks with Batch Normalization	214,442	.419	77%

Table 1

The highest accuracy is achieved by model 6 with an accuracy of 76%. Models 3 and 5 also performed well with accuracies of 71% and 73%, respectively while models 1 and 4 have the same accuracy of 66%, which is slightly lower than the average accuracy of all models. Model 2 has the lowest accuracy of 63%.

3.2 Analysis

Model 1, being a basic simple model, may not have enough complexity to capture the underlying patterns in the data, resulting in a relatively low accuracy but better than chance. Model 2, which uses depth scaling, increased the depth of the neural network with a 30-fold increase of the number of parameters to capture more complex patterns in the data. However, increasing the depth of the network did not improve the accuracy score. Model 2 also might have shown signs of overfitting since it had lower average mini-batch loss after training compared to the base model. Model 3, which uses width scaling with a 160-fold increase of the number of parameters. This increased capacity led to a higher accuracy (5% compared the base model) but the computational cost for this model needs to be considered.

Although increasing huge number of parameters in model 3 showed very low average mini-batch loss after training, and improved accuracy score, model 4 performance results suggest different. Model 4, which is implemented with compound scaling and about 61 million of parameters, achieved a lower average mini-batch loss, but similar accuracy score to model 1.

Model 5, which uses three basic residual blocks, showed increase in the network's ability to learn the underlying patterns with only double the number of parameters of model 1 but a 10% increase in accuracy. Model 5 outperformed all of the previous models. Slight modifications to model 5 such as applying batch normalization after each convolution, which is the model 6, showed further improvement in the accuracy and lower average mini-batch loss after training.

The results have shown that the shortcut connections or residual connections, work in deep neural networks like ResNet because they provide a way for the network to effectively propagate gradients during the backpropagation step of training. This is particularly important in deep neural networks where the gradients can become vanishingly small as they propagate through many layers and this method makes it easier for the network to learn complex features and converge to a good solution during training, while also helping to prevent overfitting.

Applying batch normalization to model 5 also improved the training process, which suggests this ResNet architecture has not exhibited its full potential capacity to learn the data.

4 Conclusion

By implementing simple ResNet, this study has shown that the method significantly improved accuracy in image recognition task on CIFAR-10 dataset compared to traditional deep neural networks that use depth or width scaling, as well as combined scaling. This is achieved with a smaller number of parameters, making ResNet a more efficient and effective architecture.

However, while ResNet has proven to be a highly effective architecture, there is still room for further research into hyperparameters such as learning rate, batch size, and weight decay as well as other optimization techniques such as batch normalization. The performance of the last model suggests applying various optimization techniques and fine-tuning these hyperparameters may further improve the performance of ResNet and help to uncover new insights into the inner workings of deep neural networks.

References

Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun. 2016. *Deep Residual Learning for Image Recognition*. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).