Deep Learing - Mini Project ResNet

Completed by: Anurag Mishra (aa9279), Shrey Jain(sj3396), Allen Zhang(zz3904)

This is our Github link: Codebase.

Abstract

Deep learning for image classification has exploded in popularity recently thanks to its computationally intensive, meticulously accurate neural network models. In this study, we conduct experiments using the ResNet-18 architecture with no more than 5 million model parameter restriction on the CIFAR-10 image data set. To increase the precision of our limited model, we employ hyper-parameter tuning, optimizer experimentation, data augmentation, and knowledge distillation.

Overview

Despite being more challenging to train, deep neural networks show strong performance on image classification tasks. In comparison to similar neural networks, deep residual networks (ResNets) can improve accuracy and speed up the training process. In order to achieve high test accuracy on the CIFAR-10 image classification dataset with no more than 5 million parameters, this research created a bespoke residual network (ResNet) architecture.

ResNets are made up of N Residual layers, each of which contains one or more residual blocks. Residual blocks are made up of two convolutional layers connected by a skip link. The skip connection in figure 1 is the link that skips some layers of the model.

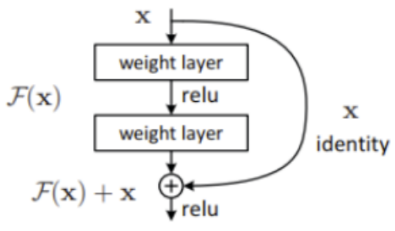


Figure 1
Residual Block
Source: Deep Residual Learning for Image Recognition [1]

The output of the model after using skip connection is:

$$H(x) = f(x) + x$$

The performance of a vision model is a product of the architecture, training methods and scaling strategy. We utilized three common techniques to achieve high accuracy for our customized residual network achieved with constraints on a number of parameters. First, we applied hyperparameter tuning to adjust the number of ResNet layers. Second, we used more data with data augmentation to train our model. Third, we trained our model with more epochs.

Problem That ResNet Solved

Researchers observed in CNNs, that after some depth, the performance degrades. This problem is relieved with Residual blocks and the ResNet model is built up of these blocks. A residual block contains two convolution layers with a skip connection from the block's input to the block's output. Every layer of ResNet is composed of several blocks. It is so because when ResNets go deeper, they normally do it by increasing the number of operations within a block. An operation here refers to a convolution of batch normalization and a ReLU activation to an input, except the last operation of a block, that does not have the ReLU. One of the problems ResNets solve is the famously known **vanishing gradient problem**.

Vanishing gradients lead to the weights never updating their values (due to sigmoid function), resulting in no learning performed. With ResNets, the gradients can flow directly through the skip connections backward from later layers to initial. Our custom architecture is illustrated in Figure below

Summary

The hyper-parameters that we vary to get the most accurate Resnet model:

N := Number of layers
B_i :=Number of residual blocks in Residual layer i
C_i :=Number of channels in Residual layer i
F_i :=Convolutional kernel size in Residual layer i
K_i :=Skip connection kernel size in Residual layer I
P := Average pool kernel size

We built our models with 4 Convolution layers and trained on the CIFAR-10 dataset to classify images. The best accuracy we achieved was 93.75% with our model using 4,935,242 parameters. The values of the hyperparameters are shown below:

Learning Rate	0.01
Weight Decay	0.0001
Optimizer	Adadelta
Scheduler	SGD with Warm Restarts
N (Number of Layers)	4
B (Number of Blocks)	3, 3, 2, 3
C (number of Channels)	64, 128, 128, 256
K (Skip Kernet)	1, 1, 1, 1
F (Cov Kernel)	3,3,3,3
P (Pool Kernet)	4

Methodology

We first loaded the CIFAR-10 dataset and split it into train, test, and validation sets. We applied two data augmentation techniques including horizontal flipping and random crop on our training set. We then tried various ResNet Hyperparameters, kernel sizes, optimizing methods and number of epochs.

ResNet Hyper-parameters

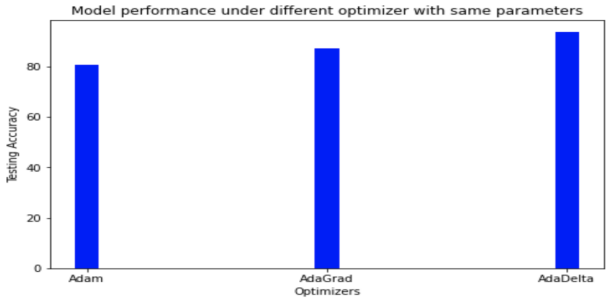
K(skip kernel), F(Conv Kernel) and P(pool kernel) the values for this hyperparameters were adapted similar to the one from [4] Wide Residual Networks paper. For the number of Layers N and Residual Blocks B, we first tried a 3-layer ResNet model with configuration of residual blocks of 1,2, and 1. Then, we trained and tested this model with varying numbers of residual blocks, recording our findings. Our findings show that the optimum results come from four layers of network.

Data Augmentation

- Random Crop This technique creates a random subset of an original image. Thus model generalization improves because the object(s) of interest we want our models to learn are not always wholly visible in the image or at the same scale in our training data. Cropping an image at multiple places augments data and also allows focusing on different areas of the image. Therefore, one is not just analyzing a picture globally but locally too using differently sized windows.
- Random Horizontal Flip This technique flips an image horizontally with a probability p. The intuition behind flipping an image is that an object should be equally recognizable as its mirror image.

Data Transformation

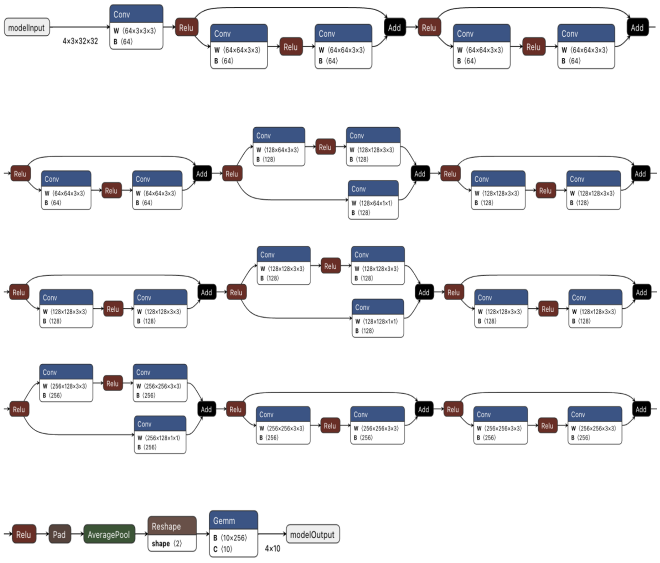
- To Tensor: It converts PyTorch tensor and scales the image by 255.
- Normalize: It transforms the images such that the values, mean and standard deviation of the image become 0.0 and 1.0 respectively. In the CIFAR-10 data set, the mean values according to respective channels are 0.4914, 0.4822, 0.4465 and the standard deviation values are 0.2023, 0.1994, 0.2010



Choice of Optimizers and Learning Rate

- An optimization algorithm finds those values of the model parameters that minimize the errors when mapping inputs to outputs. Optimization algorithms or optimizers help improve the accuracy of the deep learning model by reducing the overall loss. Several popular optimizers are:
- Stochastic Gradient Descent [16]: In Stochastic Gradient Descent (SGD), instead of taking the whole data set for each iteration, we randomly select the batches of data, then randomly shuffle the data at each iteration to reach an approximate minimum, subject to the values of the parameters momentum w and learning rate .
- Adaptive Gradient Descent: Adaptive Gradient Descent (Adagrad) uses different learning rates for each iteration. The change in learning rate depends upon the difference in the parameters during training, where more the parameters change, minor are the changes in the learning rate. Adagrad abolishes manual modification of the learning rate and converges at a comparably faster rate.
- AdaDelta : AdaDelta is a more powerful variant of the AdaGrad optimizer. It is based on adaptive learning and was created to address fundamental shortcomings of AdaGrad and RMSprop optimizers. While they require manual learning rate setup and suffer from decaying learning rate, AdaDelta stores the leaky average of the second moment gradient and a leaky average of the second moment of change of parameters in the model in two state variables.
- Adam : Adam is an extension of Stochastic Gradient Descent, where it updates the learning rate for each network weight individually. Adam inherits features of both the Adagrad and RMSprop optimizers. In Adam, we use the second moment of the gradients and calculate the mean of the uncentered variance using these moments. Adam tends to focus on faster computation time, whereas algorithms like stochastic gradient descent focus on data points.

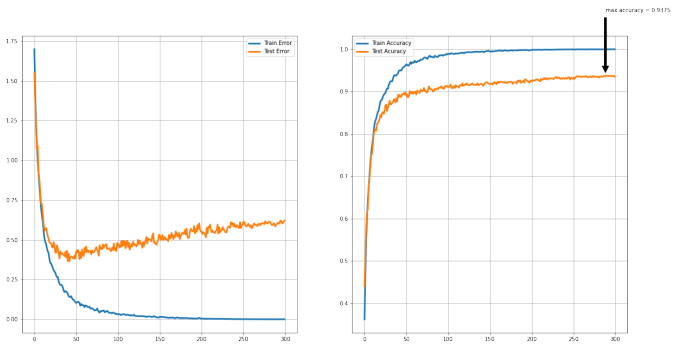
- Rectified Adam : Rectified Adam (RAdam), based on Adam optimizer, introduces a term warmup to rectify the variance of the adaptive learning rate and tackle the poor
- We tested our models on three different optimizers namely adadelta, adagrad, adam. The accuracies that we achieved are as shown in graph. These accuracies are the best accuracies for a given optimizer with varying learning rates. We start with a learning rate that decays as the number of epochs increases. Conducting training for 300 epochs, at the learning rate of 0.01 and weight decay of 0.0001. This is called Learning Rate Scheduler. Learning Rate Scheduler helps converge faster with higher accuracy. There are two types of Learning Rate Schedules - Step-wise Decay, and Reduce on Loss Plateau Decay. We use Step-Wise Decay for our model.



- We have also used Cosine Annealing technique. Cosine Annealing a type of learning rate schedule that has the effect of starting with a large learning rate that is relatively rapidly decreased to a minimum value before being increased rapidly again. The resetting of the learning rate acts like a simulated restart of the learning process and the re-use of good weights as the starting point of the restart is referred to as a "warm restart" in contrast to a "cold restart" where a new set of small random numbers may be used as a starting point.

Result

In this paper we experiment with different hyper-parameters for ResNet architecture and apply data augmentation techniques to classify images in the CIFAR-10 dataset. We achieved an accuracy of **93.75%** after training the dataset for over 300 epochs using adadelta optimizer and learning rate scheduler starting at 0.01. Our model has 4 convolutional layers with 3, 3, 2, 3 residual blocks in each layer



respectively. As restricted our experiment to just five optimizers, we believe that using more advanced optimizers and data augmentation techniques, we can achieve an even better accuracy.

References

[1] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: CoRR abs/1512.03385 (2015). arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385.

[2] Pytorch CIFAR Github. URL: https://github.com/kuangliu/pytorch-cifar.

[3] PyTorch Documentation. URL: https://pytorch.org/docs/stable/index. html.

[4] Sergey Zagoruyko and Nikos Komodakis. "Wide Residual Networks". In: CoRR abs/1605.07146 (2016). arXiv: 1605.07146. URL: http://arxiv.org/abs/1605.07146.

[5] Liyuan Liu et al. "On the Variance of the Adaptive Learning Rate and Beyond". In: CoRR abs/1908.03265 (2019). arXiv: 1908.03265. URL: http://arxiv.org/abs/1908.03265.

[6] Why perform Data Augmentation. URL: https://research.aimultiple.com/ data-augmentation/.

[7] Resnets Explained. URL: https://towardsdatascience.com/understanding- and-visualizing-resnets-442284831be8.

[8] CIFAR 10 Description. URL: https://www.cs.toronto.edu/kriz/cifar.html.

[9] Matthew D. Zeiler. "ADADELTA: An Adaptive Learning Rate Method". In: CoRR abs/1212.5701 (2012). arXiv: 1212.5701. URL: http://arxiv.org/abs/1212.5701.

[10] Learning Rate Schedules and Adaptive Learning. URL : https://towardsdatascience.com/learning-rate-schedules-and-adaptive-learning-rate-methods-for-deep-learning-2c8f433990d1