ResNet5M-CIFAR10: Image Classification

ECE-GY 7123: Deep Learning

Navoday Borkar(nyb2005), Venkata Amith Palacherla (vp2201), Jagennath Hari (jh7454)
GitHub Link : https://github.com/jagennath-hari/ResNet5M-CIFAR10.git

Abstract

Deep Learning is an emerging field of computational science that involves a large quantity of data for training a model. In this project, we have developed, programmed, and trained a ResNet architecture on CIFAR-10 dataset. The main goal of this project is to come up with an architecture having the highest test accuracy on the CIFAR-10 image classification dataset, under the constraint that model has no more than 5 million parameters. It is shown that Batch Normalization is not only important in improving the performance of the neural networks, but are essential for being able to train a deep convolutional networks. In this work various optimizers are compared on a standard dataset, CIFAR-10 with batch normalization were trained. We investigated the effects of several hyperparameters on the model's performance. We have also experimented with using data augmentation and normalization approaches to broaden the diversity of our training dataset. The final implemented architecture is effective in both memory usage and cost computation.

Introduction

A residual neural network (ResNet)[1] is an artificial neural network (ANN). It is a gateless or open-gated variant of the HighwayNet[2], the first working very deep feedforward neural network with hundreds of layers, much deeper than previous artificial neural networks. Skip connections or shortcuts are used to jump over some layers. The performance of the model drops down with the increase in depth of the architecture known as the degradation problem. In the case of ResNets, skip connections solved the degradation problem [1]. Typical ResNet models are implemented with double- or triple- layer skips that contain nonlinearities (ReLU) and batch normalization in between. Models with several parallel skips are referred to as DenseNets[3]. In the context of residual neural networks, a non-residual network may be described as a plain network.

The formulation of $F(x) + x$ can be realized by feedforward neural networks with "shortcut connections" (Fig. 1). Shortcut connections or skip connections [4] are those skipping one or more layers. In our case, the shortcut connections simply perform identity mapping, and their outputs

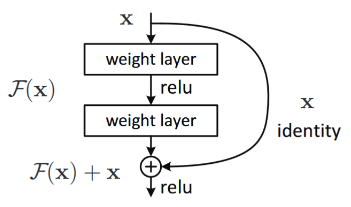
x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 1: Skip connection

are added to the outputs of the stacked layers (Fig. 1). Identity shortcut connections add neither extra parameters nor computational complexity. The entire network can still be trained end-to-end by any optimizer with backpropagation and can be easily implemented using common libraries without modifying the solvers.

The CIFAR-10 image dataset was classified using the model with the restrictions of 5 million trainable parameters. The project report intends to discuss the search for an architecture, the approach for choosing data augmentation techniques, and the effects of changing various hyperparameters to obtain a model with the highest test accuracy.

Methodology

Making the Data Loader

A Data Loader facilitates accessing and moving your data from multiple sources into a central location. In the case of Deep Learning, this often relates to segregating into mini-batches and moving them from the CPU to a single or multiple GPUs. Data Loaders having different mini-batch sizes and their influence on a ResNet were analyzed. It has often been inferred in the Deep Learning community that having a smaller batch size reduces training time, but its influence on any Neural Network is unclear.

Modern Deep Learning techniques involve augmenting each batch's data to boost performance. In our case, this involved performing multiple augmentations to our CIFAR-10 ($32 \times 32 \times 3$) images to bolster the accuracy of the trained model.

Firstly, a few images were randomly cropped, having a kernel size of 4, and this augmentation technique, assisted by flipping a few images horizontally, has improved the classification model's accuracy. Thirdly, images were normalized to change the pixel values intensity values to reinforce the classification model's performance. Lastly, the experimental auto augmentation policy offered by PyTorch for the CIFAR-10 Data set, which entailed specific data augmentation strategies such as blurring, sharpening, and brightness adjustment for this specific data set, was used to maximize classification model's accuracy .

The final phase involved splitting our data set into training and testing sets so that the neural network could be analyzed unbiasedly. Since the official CIFAR-10 provides train and test data sets, they were used respectively for the Data Loaders. For obvious reasons, no data augmentations were performed on the test set other than image normalization.

Building the model

In classical Machine Learning, the Neural Networks Hyperparameters are those parameters whose value influences the training process.

Due to the limitations set by the problem, changing the Hyperparameter values influenced the total number of parameters of the Classification Model, which had to be ensured did not cross the set limit.

Specific parameters influenced the Classification network accuracy more than others. Listing from the most influential to least influential is given below:-

1. **Number of Residual Blocks in the each Residual layer**
   There were many architectural designs in the original ResNet paper, and increasing the number of Residual blocks in each Residual layer improves the model's performance was mentioned in the paper. The original author's findings led us to try different numbers of Residual blocks to evaluate their influence on the more case-specific CIFAR-10 classification problem.
2. **The number of channels in each Residual layer**
   It is a well-known fact in the Deep Learning community that the higher number of channels in the initial layer allows us to extract more raw data from images. However, since the original image is quite noisy, the initial channels have the least values, and each layer has an increasing number of channels to extract more critical information. By keeping the initial layer's channel size to 64 and changing the magnitude of increase for each Residual layer, a more thorough analysis allowed us to achieve better accuracy during classification.
3. **Type of block in each Residual layer**
   The authors of the ResNet paper proposed an alternative to the Basic Residual block[BB], the Bottleneck Residual block[BN]. The idea is to make residual blocks as thin as possible to increase the depth and have fewer parameters, and this was accomplished by utilizing 1x1 convolutions to create a bottleneck operation. Although it was stated that bottlenecking is effective on higher-resolution image inputs, its influence on the overall accuracy of the trained network was carefully analyzed.
4. **Convolutional Kernel Size in Residual Layer**
   The problem included this as a Hyperparameter, but its influence was unclear. However, testing was carried out to comprehend the effect on the Classification problem.
5. **Skip connection kernel size in Residual layer**
   Through testing was conducted on this Hyperparameters, it had little influence on the final accuracy.
6. **Average pooling Kernel Size**
   Average pooling is critical for final accuracy and not changed as it may increase the number of false positives in the trained network.

Testing different optimizer and schedulers

Optimization algorithms are responsible for mapping training loss with attributes of Neural Network such as weights and learning rates in order to reduce the losses, and make the model more accurate. During training for each batch loss is accumulated and weights of the model are updated based on the accumulated loss. Each optimizer has a specific strategy for updating the weights using the loss. Different optimizers used to find the best performing model are -

Stochastic gradient descent(SGD) with momentum and weight decay

Vanilla SGD involves updating weights using the negative gradient of loss after training each batch instead of whole dataset. This help in faster convergence. Equation of SGD is given by-

$$W = W - \alpha * \nabla L(W)$$

Where α is the learning rate.
Downside of Vanilla SGD is noisy weight updates due to the variance in among the batches. To solve this SGD with momentum is used where moving average of gradient descents are used to update weights. Along with this to remove the bias of few weights on the output weight delay factor is used to update weights. General weight update equation is given by -

$$v_t = \beta * v_{t-1} + \alpha * \nabla L(W)$$

$$W = W - v_t - \alpha * \frac{\lambda * W}{2}$$

where λ is a factor for decreasing weights.

Adaptive gradient Optimizer(Adagrad)

Adagrad is a optimizer where learning rate is not fixed for all the weights and it depends on current and past gradients of that weight. Learning rate is inversely proportional to sum loss gradients of the weight computed till that time step. When sum of gradients are large update in weights are small. Weight update equation is given by -

$$w_{t+1,i} = w_{t,i} - \frac{\eta}{\sqrt{G_{i,i} + \sigma}} * \nabla L(w_i)$$

where G_t is a diagonal matrix where diagonal matrix where each diagonal element (i, i) is the sum of the squares of the gradients w.r.t. w_i up to time step t.

RMSProp gradient

RMSprop uses a moving average of squared gradients to normalize the gradient. This normalization balances the step size (momentum), decreasing the step for large gradients to avoid exploding and increasing the step for small gradients to avoid vanishing. Update equations of RMSProp is given by-

$$v = \beta * v + (1-\beta) * (\nabla L(W))^2$$
$$W_t = w_{t-1} - \frac{\eta}{\sqrt{v}+\sigma} * \nabla L(w_i)$$

Adam Optimizer

Adam optimizer combines of both first and second momentum of loss derivatives to update the weights. Weights update equation are given as -

$$m_t = \beta_1 * m_t + (1-\beta_1) * (\nabla L(w_i))$$
$$v = \beta * v + (1-\beta) * (\nabla L(W))^2$$
$$w_t = w_{t-1} - \alpha * (\frac{m_t}{\sqrt{v_t}+\eta})$$

Schedulers

Learning rate is one of the important hyper-parameter used in training Neural Network. Schedulers change the learning rate with number of iteration in the training. It is seen in the research that having a bigger learning rate at the start and smaller learning rate towards the end of training helps in faster convergence.
Following two schedulers are used while training different models -

One-cycle learning rate

This scheduler slowly increases the learning rate in the initial duration of the training and decreases it down in the final duration of the training, and parameters for momentum are decreased and increased respectively.

Cosine Annealing

This is 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.

Results

Models were trained using cross entropy loss with a maximum learning rate of 0.01 for 200 epochs. Following sections discuss the various experiments carried out -

Effect of accuracy with change in number of Convolutional channels in each layer

The number of channels is the decisive factor in attaining high accuracy on the CIFAR-10's test data set. After careful evaluation, it was determined that having higher channels at each layer guarantees a good performance. On one such occasion, a model trained using a lower number of channels had an average accuracy on the test data set of less than 90%. It was noted that a channel size greater than 200 at the final layer dramatically increases accuracy and guarantees it greater than 90%.

Effect of accuracy with change in number of Residual Blocks in each layer

The CIFAR-10 data set has lower resolution images and having deeper networks did not perform better than the network with less layers. Training attempts were made using ResNet 18, 56, 101 and 156 architectures. Few architectures of Resnet 18, 56, and 156 layers gave similar accuracy greater than 95%. By carefully reducing the number of channels in each Convolutional with increase in depth of the network ensured the total number of parameters did not exceed 5 Million.

Residual Block Style

Each layer in the model was tested using the standard Residual Block, Bottleneck Block or a combination of both at each layer. After evaluating each style and combination of both, it can be safely said that the standard Residual Block is the best. Using a combination of both also achieves similar results. Moreover, the standard block also uses fewer parameters, so this ended up being the favorite. It was stated by the authors that the Bottleneck Block works best on deeper networks and since our input images being lower resolution and restrictions of 5 Million parameters causes some compatibility issues with the Bottleneck style.

Different Optimizers

Optimizers are critical to reducing the losses attained by the network during the training process, various different optimizers mentioned above. SGD with momentum(SGD+M) performed best, while Adam was a close second. The Deep Learning community frequently claim that SGD works best on classification problems, and this may be the reason why SGD performed the best.

Best model

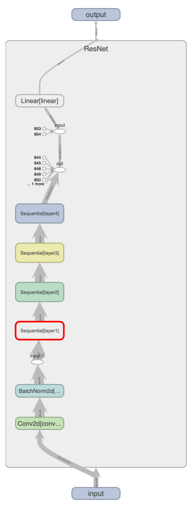

Figure 2: Best Model's architecture

S.No	Block Type	# Residual Blocks at layer	# Conv Channels at layer	Optimizer	Params	Final Acc
1	[BB, BB, BB, BB]	[2,3,2,2]	[64,128,192,256]	SGD+M	4.47M	**93.32**%*
2	[BB, BB, BB, BB]	[2,3,2,2]	[64,128,192,256]	Adagrad	4.47M	90.55%
3	[BB, BB, BB, BB]	[2,3,2,2]	[64,128,192,256]	RMSProp	4.47M	89.130%
4	[BB, BB, BB, BB]	[2,3,2,2]	[64,128,192,256]	Adam	4.47M	93.05 %
5	[BB, BB, BB, BB]	[2,2,2,2]	[64, 128, 232, 268]	Adam	4.99M	81.03%
6	[BB, BB, BB, BB]	[2,2,2,2]	[64, 128, 232, 268]	SGD+M	4.99M	**95.55**%
7	[BN, BN, BN, BN]	[2,2,2,2]	[64, 118, 178, 256]	SGD+M	4.99M	91.97%
8	[BN, BB, BB, BN]	[2,2,2,2]	[64, 128, 232, 256]	SGD+M	4.99M	92.39%
9	[BB, BN, BN, BB]	[2,2,2,2]	[64, 100, 160, 256]	SGD+M	4.99M	92.73%
10	[BN, BN, BN, BN]	[3,4,6,3]	[64, 96, 128, 190]	SGD+M	4.98M	**95.51**%
11	[BN, BN, BN, BN]	[3,4,6,3]	[64, 96, 128, 190]	Adam	4.98M	93.37%*
12	[BN, BN, BN, BN]	[3,8,36,3]	[64, 72, 74, 90]	SGD+M	4.99M	**93.82**%*

Table 1: Trained model(s) summary[* Trained for only 130 epochs][BB = Basic Block, BN = Bottle Neck]

Our best model(6) had 4 Residual layers, and each had 2 Residual blocks in them. The initial Convolutional channels in each Residual layer were $64, 128, 232, 268$, and the standard Residual Block was preferred over the Bottleneck Block due to the image resolution being low on the CIFAR-10 data set. Using a Cosine Annealing scheduler and SGD as the optimizer with momentum and weight decay, the accuracy of the Classification model was bolstered. Few models (1,10,12) had the potential to outperform the best model(6) but due limitation in hardware they weren't trained for more than 140 epochs.

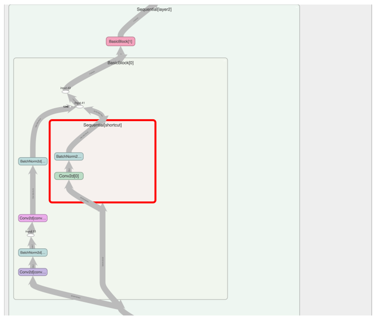

Figure 3: Best Model's skip connection

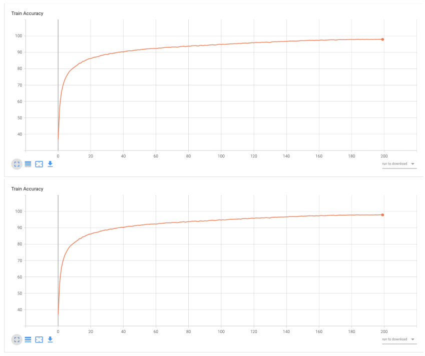

Figure 4: Best Model's Train Loss and Train Accuracy

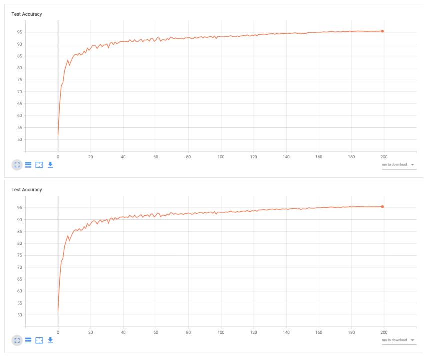

Figure 5: Best Model's Test Loss and Test Accuracy

Conclusion

By aggressively changing the Hyperpameters and optimizers, the final model under 5 Million parameters and using SGD+M achieved an accuracy of 95.55% on the CIFAR-10's test data set. It was seen that either increasing channels of shallow networks or increasing depth and decreasing the channels gave similar results.

References

[1] He, Kaiming and Zhang, Xiangyu and Ren, Shaoqing and Sun, Jian *Deep Residual Learning for Image Recognition*, 2015, https://doi.org/10.48550/arxiv.1512.03385

[2] Srivastava, Rupesh Kumar and Greff, Klaus and Schmidhuber, Jürgen Highway Networks, 2015, https://doi.org/10.48550/arxiv.1505.00387

[3] Huang, Gao and Liu, Zhuang and Van Der Maaten, Laurens and Weinberger, Kilian Q. *Densely Connected Convolutional Networks*, 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[4] Bishop, Christopher M and others Neural networks for pattern recognition, 1995, Oxford university Press.