Deep Learning Mini Project Report

Jinrui Zhang
jz6578
Code: https://github.com/jinrui-rey/DL_MiniProject

Abstract

This project presents a modified ResNet architecture designed specifically for optimizing performance on the CIFAR-10 image classification dataset. Through exhaustive experimentation with diverse parameters and hyperparameters, the achieved accuracy reached a notable 92.65%. Also, this project highlight how these parameters influence the performance of the network. However, owing to computational limitations, the network's performance potential may not have been fully realized. It is anticipated that with increased iterations and a larger batch size, the ResNet model developed in this study could achieve even higher accuracy.

Overview

This project endeavors to enhance performance on the CIFAR-10 image classification dataset by employing a modified residual network (ResNet) with fewer than 5 million trainable parameters. The architecture of the ResNet developed in this study is depicted in Figure 2, with further details of this network provided later in the report.

During experimentation, the decay rate of the learning rate was not treated as a variable. However, based on insights gained from network construction and preliminary trials, a decay rate of 0.2 was selected to optimize network performance.

Through rigorous parameter exploration—including batch size, learning rate, decay step, and optimization strategy—it was determined that stochastic gradient descent (SGD) is the most effective strategy when other parameters remain constant. While acknowledging that alternative optimization strategies may excel with appropriately tuned learning rates and corresponding parameters, this project exclusively employs SGD as the optimization strategy. Notably, the model achieves peak performance with a learning rate of 0.3, facilitating rapid convergence during iterations. The learning rate decay step is approximately 8000, and the batch size is set as high as the computing device allows, ranging from 32 to 128. By optimizing these variables accordingly, the model achieves its highest accuracy of 92.65

Methodology

Dataset

The dataset applied in this project is the CIFAR-10 image classification dataset. The CIFAR-10 dataset is a popular benchmark dataset for image classification tasks in the fields of computer vision and machine learning. It consists of 60,000 32×32 pixels colored images in 10 classes, with 6,000 images per class. The dataset is divided into 50,000 training images and 10,000 test images. Each image in the dataset is associated with a single class label (an integer from 0 to 9), indicating the category of the object depicted in the image. Figure 1 shows a few samples in the CIFAR-10 dataset.

Figure 1: Sample image

Model Architecture

The overview of network architecture is shown in Figure 2. There are 4 residual layers; each layer has 3, 6, 4, and 3 residual blocks.

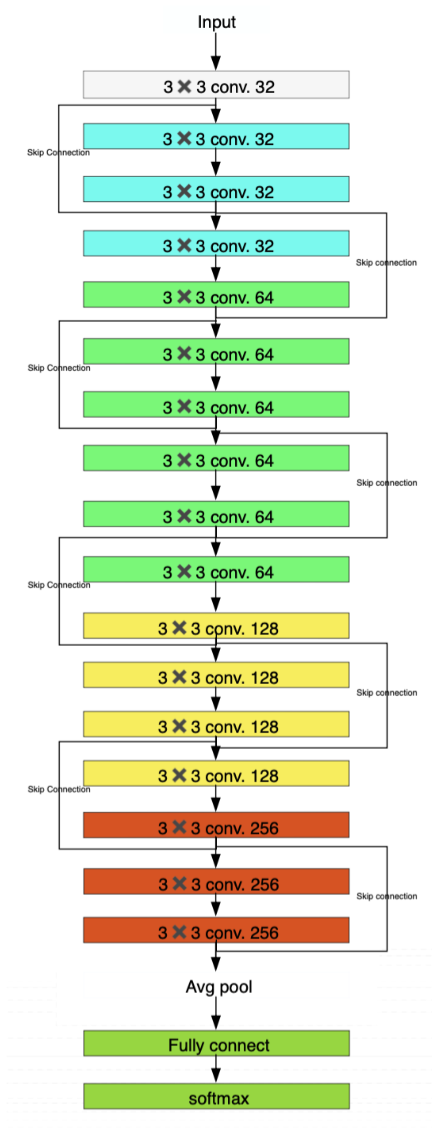

Figure 2: Architecture of ResNet

As mentioned above, the CIFAR-10 dataset has images that are 32×32 pixels in size and 3 channels. The input image needs to pass through a convolutional network and batch norm layer before accessing the first residual layer. In this convolution network, the kernel size is 3, the stride is 1 and the padding is 1. After the initial convolutional layer, the channel changes from 3 to 32. Therefore, the input channel of the first residual layer is 32.

Under deep consideration and a large number of trials, with both constraint on the total number of trainable parameters and the performance of the network, the number of residual blocks in each layer is set to 3, 6, 4, and 3. The number of blocks in each layer will not be treated as variable in this project. This setting will lead the total number of trainable parameters to 4.887 million.

The first residual layer consists of 3 residual blocks. Figure 3 shows the architecture of the residual block. The first residual layer has 32 input channels, and the kernel size is 3. The skip connection kernel size is 1. The rest of the residual layers each consist of 6, 4, and 3 residual blocks, which are exactly the same as shown in Figure 3. The input channel of the second residual layer is 64, 128 for the third layer and 256 for the fourth layer. And all these three layers have the same kernel size of 3 and stride of 2. The average pool kernel size is set as 4. All hyper-parameters is shown in Table

hyper-parameters	value
Residual layers	4
Residual blocks in layer i	3 (layer1) 6 (layer2) 4 (layer3) 3 (layer4)
Channels in layer i	32 64 128 256
Conv. kernel size in each layer	3
Skip connection kernel size	1
Average pool kernel size	4

Table 1: hyper-parameters

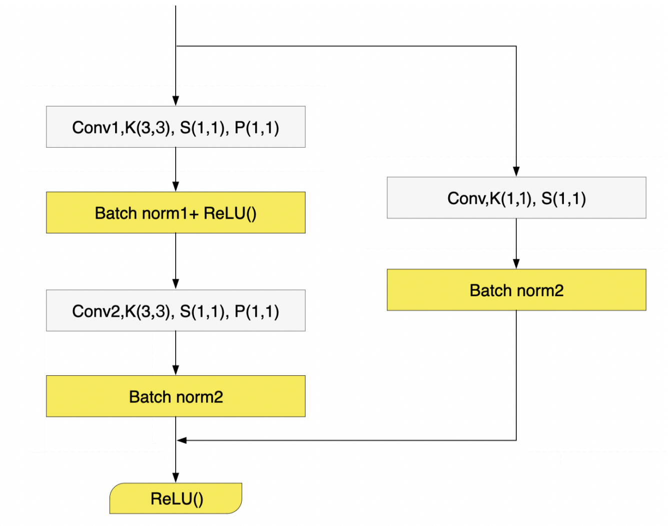

Figure 3: Architecture of residual block

Loss function

The chosen loss function for this project is the Cross-Entropy function. Cross-entropy is employed to assess dissimilarities between two probability distributions, specifically comparing the true distribution of labels with the predicted distribution derived from the model. This metric quantifies the alignment of predicted probabilities with the actual labels, aligning seamlessly with the objectives of this project.

Optimization strategy

The optimization strategy applied in this project is treated as a variable since one of the goals of this project is to find out which optimization strategy fits well. There are four different choices of optimization strategies: stochastic gradient descent (**SGD**), Adaptive Moment Estimation (**Adam**), Adaptive Gradient Algorithm (**Adagrad**), and Accelerated Stochastic Gradient Descent (**ASGD**).

Each of these optimizer has specific advancement:

SGD: SGD updates model parameters using random mini-batches of training examples, which introduces noise into the gradient estimates. This stochastic sampling allows for faster training and convergence, especially on large datasets.

Adam: Adam is well-suited for training deep neural networks due to its ability to converge quickly and handle noisy gradients effectively.

ASGD: This optimization strategy is well-suited for distributed training settings, where data is distributed across multiple machines or processors. It supports parallel updates and synchronization of model parameters, making it scalable to large datasets and computational resources.

Adagrad: This optimizer scales the learning rates inversely proportional to the square root of the sum of squared past gradients for each parameter. This rescaling helps handle sparse gradients and different parameter scales effectively.

Experiment

In this study, the experiment involves the systematic variation of key parameters including learning rate, optimization strategy, decay step, and batch size. The specific configurations of these variables across different experiments are documented in Table 2.

The experimental approach employed in this study is the controlled variable method, where variations in one parameter are examined while keeping other parameters constant. Specifically, changes to individual parameters are investigated under controlled conditions, ensuring that any observed effects can be attributed to the manipulated variable while minimizing confounding factors from other experimental variables. As the experiment progresses, the parameters that resulted in optimal network performance in the previous experiment will be kept fixed, while attention shifts to varying other parameters. This iterative process aims to achieve the best overall performance of the network by systematically optimizing each parameter in turn.

Variables	Choices			
Optimizer	Adam	SGD	Adagrad	ASGD
learning rate	0.1	0.3	0.5	-
decay step	4000	5000	8000	10000
batch size	32	64	128	256

Table 2: variable setting

The learning rate decay parameter holds significant influence within this project. However, constrained by computational resources, the decay rate will remain fixed across experiments. Through iterative network development and preliminary trials, an optimal decay rate of 0.2 was identified as yielding superior performance. Table 3 enumerates the fixed parameters utilized in the experimentation process.

Invariables	value
decay rate	0.2
momentum	0.9
weight decay rate	0.0001
iterations	32,000

Table 3: Invariable setting

Optimizer

The evaluation of loss using different optimization strategies is depicted in Figure 4, while the corresponding final accuracy results are presented in Table 4. It is notable that **ASGD** demonstrated the highest accuracy following 32,000 iterations, whereas the **Adam** strategy exhibited the lowest accuracy. However, the lower final accuracy observed with **Adam** does not imply that it is an inadequate optimization strategy. It is plausible that **Adam** could achieve significantly higher accuracy with appropriate parameter tuning.

In summary, **ASGD** attained the highest final accuracy in this evaluation. Nevertheless, taking into account time consumption and the disparity in final accuracy between **SGD** and **ASGD**, this project will proceed with utilizing **SGD** as the chosen optimizer for subsequent experiments.

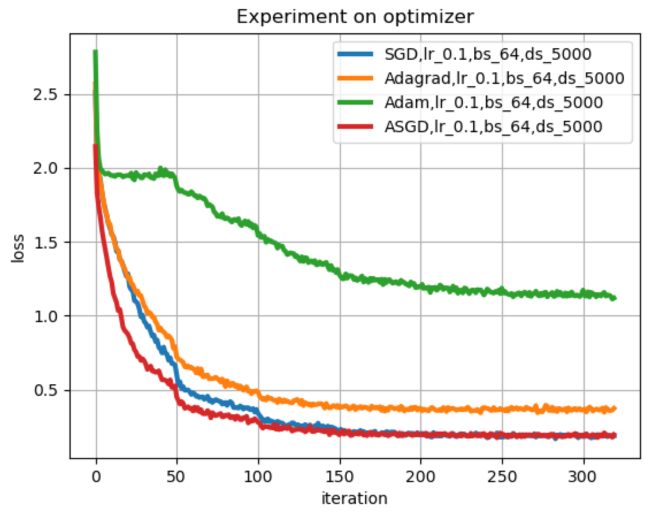

Figure 4: Loss evaluation on different optimizers

Learning rate

The assessment of loss using varying learning rates is visualized in Figure 5, and the corresponding final accuracies are

Optimizer	Final accuracy (%)
Adam	60.47
SGD	88.51
Adagrad	84.9
ASGD	**88.95**

Table 4: Final accuracy on different optimizers

detailed in Table 5. Based on the findings of this experiment, optimal performance was achieved with a learning rate set to 0.3. Therefore, for subsequent experiments, the learning rate will be fixed at 0.3.

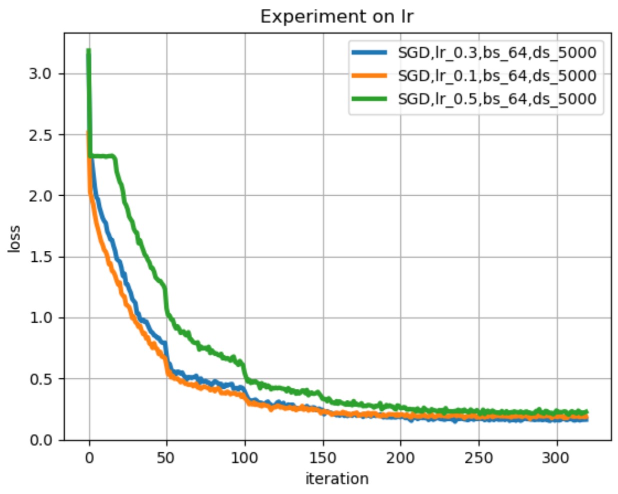

Figure 5: Loss evaluation at different learning rates

Learning rate	Final accuracy(%)
0.1	88.51
0.3	**89.31**
0.5	87.70

Table 5: Final accuracy at different learning rates

Batch size

The evaluation of loss using different batch sizes is depicted in Figure 6, while the corresponding final accuracy results are presented in Table 6. Based on the analysis of the figure and table, it is observed that the highest accuracy was achieved when the batch size was set to 128. Therefore, to ensure optimal performance of the network, the batch size will be fixed at 128 for subsequent experiments.

Decay step

The evaluation of loss using different decay steps is plotted in Figure 7, and the corresponding final accuracy results are presented in Table 7. Based on the analysis of the figure

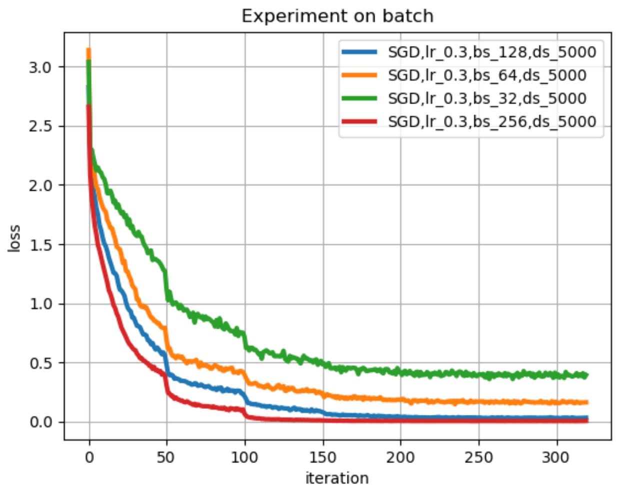

Figure 6: Loss evaluation at different batch sizes

Batch size	Final accuracy(%)
32	84.96
64	89.31
128	**91.07**
256	90.40

Table 6: Final accuracy at different batch sizes

and table, the highest final accuracy achieved was 92.65%, which is also the highest accuracy achieved in all sets of experiments. Additionally, the loss evaluation plot reveals a noticeable steep drop at each decay step. This pattern indicates that when the iteration count reaches a multiple of the decay step, the learning rate undergoes decay, reducing to 20% (since the decay rate is set at 0.2) of its current value. This reduction in learning rate facilitates better convergence towards the optimal point, enhancing the overall performance of the model.

Decay step	Final accuracy(%)
4000	89.83
5000	91.07
8000	**92.65**
10000	92.14

Table 7: Final accuracy at different decay steps

Discussion

Based on the conducted experiments, I systematically manipulated a single variable per experiment to isolate and scrutinize specific factors. As a result, the final accuracy reached 92.65%. The optimal parameter configurations are detailed in Table 8. It is anticipated that further enhancements in accuracy could be achieved by augmenting the iter-

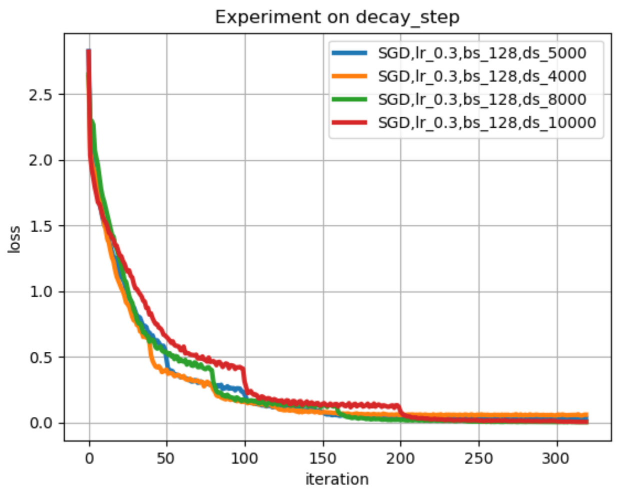

Figure 7: Loss evaluation at different decay steps

variable	value
optimizer	SGD
learning rate	0.3
batch size	128
decay step	8000

Table 8: Optimized parameter

ation count. Consequently, subsequent to refining these critical parameters, I escalated the iteration count, leading to an enhanced final accuracy of 93.18%. The corresponding experiment log file is accessible here

Conclusion

In this study, the adapted ResNet architecture developed encompasses a total of 4.884 million trainable parameters. A detailed breakdown of these parameters can be accessed via the provided link (here). This network has exhibited the capability to achieve a classification accuracy exceeding 90% when appropriately configured. Comprehensive information regarding all parameters and hyperparameters employed can be found in Tables 3, 8, and 1, while the corresponding loss evaluation plot is presented in Figure 8.

However, owing to computational capacity constraints and time limitations, the full potential of this network remains untapped. Optimizing parameters and hyperparameters such as decay rate, weight decay rate, kernel size within residual layers, skip connection kernel size, and average pool kernel size holds promise for significantly enhancing network performance.

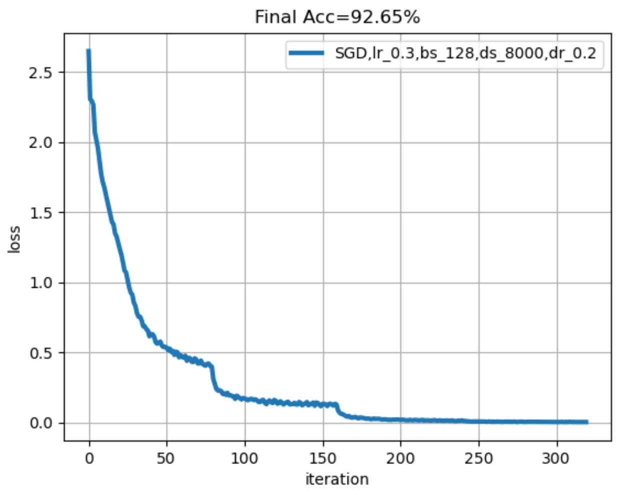

Figure 8: Best performance