# Midterm report: study neural network on CIFAR-10

Jinzhi Li[1], Siyu Yuan[2], and Hongxu chen[3]

[1] Fudan University 20110980006@fudan.edu.cn
[2] Fudan University 17307110448@fudan.edu.cn
[3] Fudan University 20110980002@fudan.edu.cn

**Abstract.** In this paper, we apply ResNet18 to achieve classification on CIFAR-10 dataset. We analyse the influence of different parameters, such as batch size, learning rate and so on, on accuracy of test set and visualized results are presented in detail.

**Keywords:** ResNet-18 · Batch Normalization · Loss Landscape · Accuracy Landscape.

## 1 Dataset

The CIFAR-10 dataset is a collection of images that are commonly used to train machine learning and computer vision algorithms[3]. It is one of the most widely used datasets for machine learning research. The CIFAR-10 dataset contains 60,000 32x32 color images in 10 different classes. As shown in Fig.1, the 10 different classes represent airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. There are 6,000 images of each class.Computer algorithms for recognizing objects in photos often learn by example. CIFAR-10 is a set of images that can be used to teach a computer how to recognize objects. Since the images in CIFAR-10 are low-resolution (32x32), this dataset can allow researchers to quickly try different algorithms to see what works. Various kinds of convolutional neural networks tend to be the best at recognizing the images in CIFAR-10.

### 1.1 Construction

We obtained the latest dataset of CIFAR-10 from its open websites[4] and the size of the dataset is 60k. Then divided them into training set, validation set and test set according to 10:1:1.

## 2 Methodology

In this section, we introduce our model in detail.

[4] https://www.cs.toronto.edu/ kriz/cifar.html

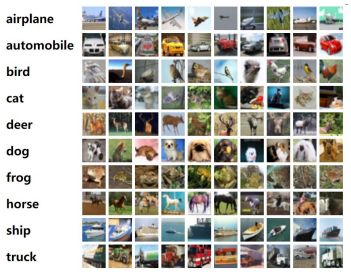


Fig. 1: The classes in the CIFAR-10 dataset, as well as 10 random images from each

### 2.1 Framework

In this task, we apply ResNet-18[1] as our basic framework. As shown in Fig.2, ResNet-18 consists of 17 Convolutional Layer and 1 fully connected layer. The novelty of ResNet-18 is that the model explicitly lets these layers fit a residual mapping rather than hoping each few stacked layers directly fit a desired underlying mapping.

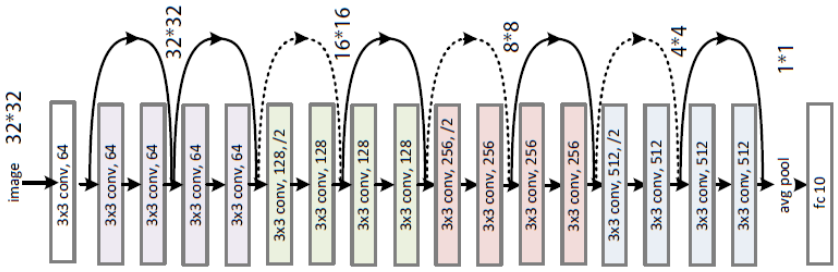


Fig. 2: The network structure of ResNet-18

## 3 Experiments

In this section, we explore the factors of our model and training process that influence the accuracy on test dataset. In order to facilitate the horizontal comparison of the influence of each factor on the experimental results, we set epoch=200. Please note that in our report, the data of loss and accuracy comes from tensorboard and we use the Matplotlib package instead of TensorBoard because Matplotlib shows the effects of individual parameters more intuitively than TensorBoard, and the graphics of Matplotlib are much more aesthetic.

### 3.1 Model Factors

**Batch Normalization** Batch Normalization (BN)[2] is a widely adopted technique that enables faster and more stable training of deep neural networks (DNNs), which is a technique that aims to improve the training of neural networks by stabilizing the distributions of layer inputs.. We analyse the influence of BN on the accuracy and loss of the test set. The results are shown in Fig.3. The effect of adding BN model is better than not adding BN. We believe that when BN is not added, the data in the middle layer tends to be extremely distributed (extreme mean or extreme variance), leading to the disappearance of the backpropagation gradient or the invalid nonlinear activation function. BN can adjust the data distribution in the middle layer to make more effective use of the nonlinear activation function with a more ideal data distribution.

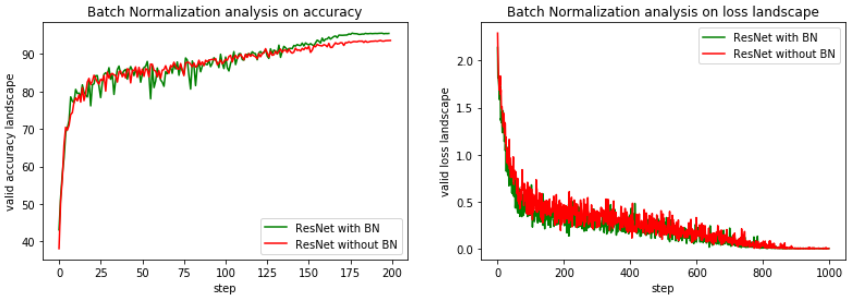


Fig. 3: The influence of BN on the test set

**Shortcut** ResNet is equipped with shortcut connections between layers, which skip layers in the forward step of an input, and exhibits efficient training using simple first order algorithms. We compared ResNet18 with shortcut and without shortcut. The results are shown in Fig.4.

From the figure, we find that shortcut can improve accuracy of test set and increase the coverage speed. Without residual connections, gradients of backpropagation may disappear in the convolution layers and activation function layers since gradients maybe set zero during the calculation. In fact, we think residual connection can serve as a highway to convey gradients, which is beneficial for training process. As a result, the training procedure becomes faster and more stable with residual connection.

### 3.2 Training Process Factors

**Optimizer** We fix Channel1=64, loss function=CrossEntropyLoss, activation function=Relu. To demonstrate the necessity of momentum in SGD, we compare

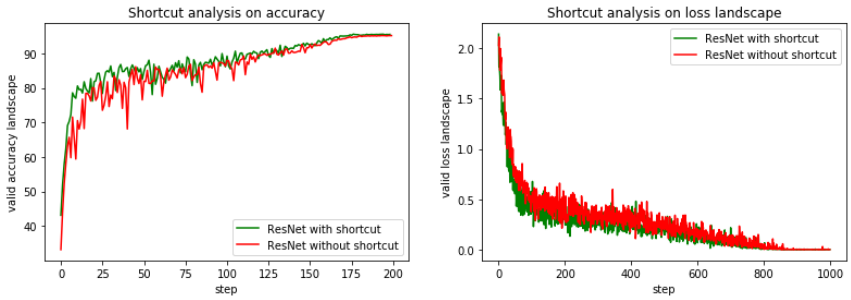


Fig. 4: The influence of shortcut on the test set

SGD with momentum and SGD without momentum. The results are shown in Fig.5.

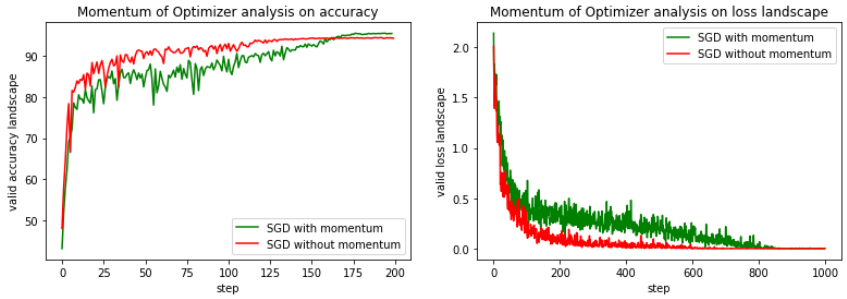


Fig. 5: The influence of momentum on the test set

Introducing the Momentum method, on the one hand, is to solve the "canyon" and "saddle point" problems. On the one hand, it can also be used for SGD acceleration, especially for high curvature, small but consistent gradients. If you think of the original SGD as a ball of paper rolling down under the action of gravity, the small mass is greatly interfered by the elasticity of the mountain wall, causing it to oscillate back and forth, or the speed at the saddle point is quickly reduced to 0 due to the small mass, making it impossible to leave the flat ground. The momentum method is equivalent to replacing the paper ball with an iron ball, which is not easily disturbed by external forces, and the trajectory is more stable. At the same time, because of the inertia at the saddle point, it is more likely to leave the flat ground. The Momentum gradient descent method

is to calculate the exponentially weighted average of the gradient and use it to update the weight. Its running speed is almost always faster than the standard gradient descent algorithm.

**Activation Function** We fix Channel1=64, loss function=CrossEntropyLoss, Optimizer = SGD and change the activation function. The accuracy results are shown in Table.1. From the table, we can find that the activation functions of relu is the best, followed by tanh, and sigmoid has the lowest accuracy.

Table 1: The influence of activation function on the test set

| Activation Function | Accuracy |
|---|---|
| Sigmoid | 87.63% |
| Tanh | 92.37% |
| **Relu** | **95.48%** |

The influence of Activation Function on the changes of accuracy and loss are shown in Fig.6. From the figure comparison of Relu, Sigmoid and Tanh, we can discover that Relu performs the best with the highest accuracy and the highest learning speed. We find that the vital reason for the slow convergence of these two activation functions is that the gradient of x will be almost zero if x is too big in both Sigmoid(x) and tanh(x).

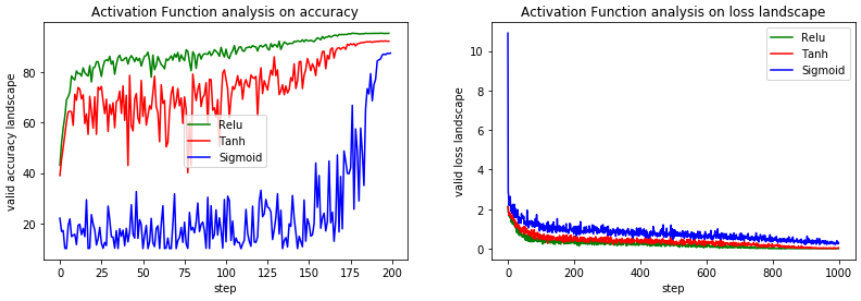


Fig. 6: The influence of Activation Function on the test set

**Loss Function** We fix Channel1=16 and activation function=Relu and change the loss function. Since our task is Multi-classification task, CrossEntropyLoss

and MultiMarginLoss can be adopted in our task. The results are shown in Table.2. From the table, we can find that CrossEntropyLoss is better than MultiMarginLoss.

Table 2: The influence of loss function on the test set

| Loss function | Accuracy |
|---|---|
| MultiMarginLoss | 95.39% |
| **CrossEntropyLoss** | **95.48%** |

The influence of Loss Function on the changes of accuracy and loss are shown in Fig.7. The reason is that CrossEntropyLoss applies a softmax to turn rating results to probabilistic result, which is much more suitable for classification tasks than MultiMarginLoss, also resulting a faster training speed. However, we also discover that in the loss landscape, MultiMarginLoss performs actually better than CrossEntropyLoss with a faster decreasing speed.

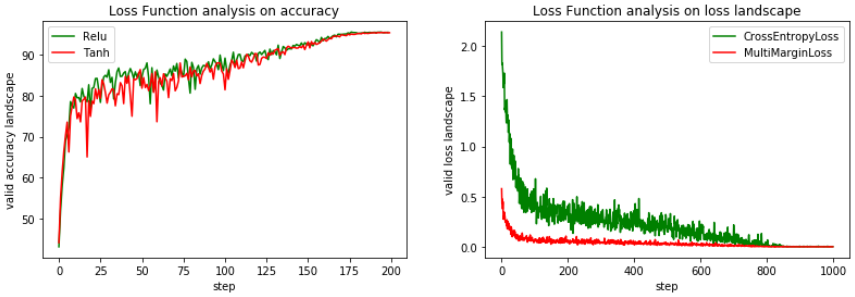


Fig. 7: The influence of Loss Function on the test set

**Learning Rate** Learning rate is a tuning parameter in an optimization algorithm that determines the step size at each iteration while moving toward a minimum of a loss function.In setting a learning rate, there is a trade-off between the rate of convergence and overshooting. While the descent direction is usually determined from the gradient of the loss function, the learning rate determines how big a step is taken in that direction. A too high learning rate will make the learning jump over minima but a too low learning rate will either take too long to converge or get stuck in an undesirable local minimum. In this task, we tried to compare the fixed learning rate with Cosine Annealing LR strategy.

Cosine Annealing LR lets the learning rate change periodically with epoch. The accuracy results are shown in in Fig.8.

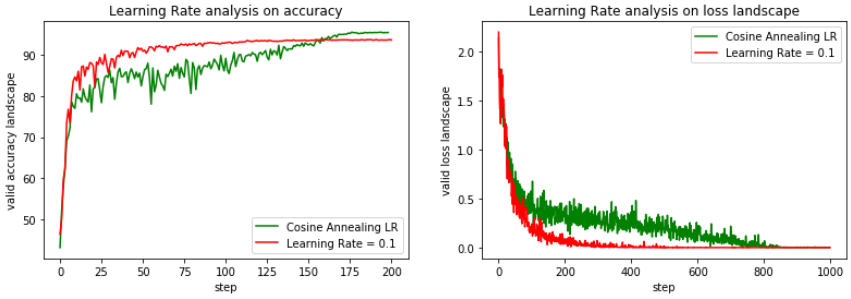


Fig. 8: The influence of Learning Rate on the test set

**Batch Size** Larger batch size means the network can update parameter values based on much more number of training images. As a result, it can lead to an increase in accuracy. The results are shown in Table.3. With increasing batch size, accuracy barely changes.

Table 3: The influence of batch size on the test set

| Batch Size | Accuracy |
|---|---|
| 64 | 94.89% |
| 256 | 95.27% |
| **128** | **95.48%** |

The influence of batch size on the changes of accuracy and loss are shown in Fig.9. As can be seen from the figure, larger batch size can achieve higher accuracy and faster convergence speed. However, due to the limitations of computer memory, the batch size cannot be too large.

## References

1. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)

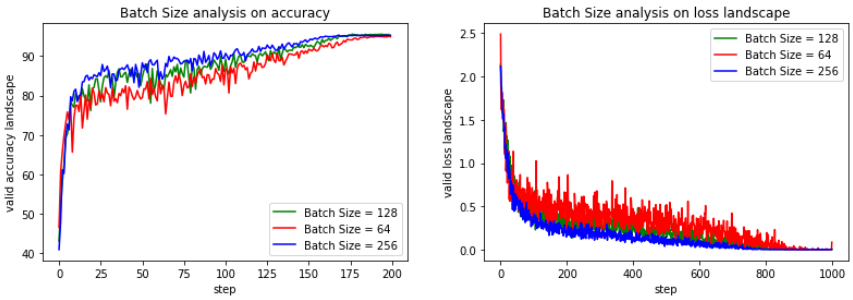


Fig. 9: The influence of Batch Size on the test set

2. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International conference on machine learning. pp. 448–456. PMLR (2015)
3. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)