Reproducibility of "ImageNet Classification with Deep Convolutional Neural Networks"

Sherry Shang
McGill University
xiaoxiao.shang@mail.mcgill.ca

Zijian Pei
McGill University
zijian.pei@mail.mcgill.ca

Yuteng Zhang
McGill University
yuteng.zhang2@mail.mcgill.ca

Reproducibility Summary

Scope of Reproducibility

The original paper [2]ImageNet Classification with Deep Convolutional Neural Networks introduces a Convolutional Neural Network Algorithm with five convolutional layers and three fully-connected layers. The authors ran the algorithm on ImageNet dataset; while we run the same algorithm on [1]CIFAR-10 Dataset, a dataset examined by the authors as well. In addition, it is being discussed in the article that the dropout method is an effective way to avoid overfitting and train the model with more robust features. Therefore, we conducted another experiment with a low dropout rate to investigate the impact. In addition, the activation function is explicitly mentioned. In general, ReLU is compared to tanh, and it provides a faster convergence rate and we will try to verify this in another experiment. Finally, we want to challenge AlexNet's architecture by looking at what happens when one layer is removed.

Methodology

We did not use code from any website or from the authors of the original paper, we followed the concept from the paper and used tensorflow and kersas packages to rebuild the model from scratch and implemented techniques like "Flatten", "Dropout" and "Conv2D".

Results

According to our reproduction, the overall accuracy of the model is 59.33%, which is close to the Top-1 error rate of 40.9% in the original paper. We claim that our result supports the original paper. Besides, here are the findings from extra experiments outside the scope of the paper. First, by reducing the dropout, there is even a decrease in the model overfitting and the accuracy also increased by 0.40%. Also, when we use tanh as the activation function, the time for reaching a 25% training error is 15 minutes slower than using ReLu. Then, we found that after reducing one convolutional layer, the model performance surprisingly gets 1.69% better on the same dataset.

What was easy

The authors used fundamental techniques of machine learning to build their optimized models, such as using ReLU as activation function and adding dropout. As a result of that, we can easily build and train our model using functions from keras and tensorflow packages. We built the overall architecture of our CNN model without many difficulty.

What was difficult

The original paper uses ImageNet to test the performance of the model, however we found that it is impossible to train ImageNet dataset in few hours due to the limitation of time and resources. Therefore, we have to find a replacement, and we chose CIFAR-10 dataset, a. dataset with the same form of image but much fewer entries and classes. CIFAR-10 dataset is used as a measure of the overall performance of the authors' models as well. Because we are applying our models on a different dataset, we have to understand every aspect and every number presented in the original paper. For example, we have to fit our model to a 10-way softmax instead of 1000 in the original paper.

Submitted to ML Reproducibility Challenge 2020. Do not distribute.

Communication with original authors

We do not have any contact with the original authors.

1 Introduction

In this project, we investigate the reproducibility of the paper "ImageNet Classification with Deep Convolutional Neural Networks". The model provided in the original paper is believed to be able to increase the accuracy of large-scale image classification problem and prevent overfitting, we managed to rebuild the model and test its performance on a smaller dataset and verify the statement and truthfulness stated in the original paper.

2 Scope of reproducibility

In addition to reproducing what the authors did in the original report, we also wanted to verify some of the claims made in the original article or made by ourselves. We will study the effect of each factor in the results.

- Claim 1: Adding dropout helps reducing over-fitting.
- Claim 2: ReLU provides faster convergence speed than tanh.
- Claim 3: Network's performance degrades if a single convolutional layer is removed.

Each experiment in Section 4 will support these claims.

3 Methodology

3.1 Model descriptions

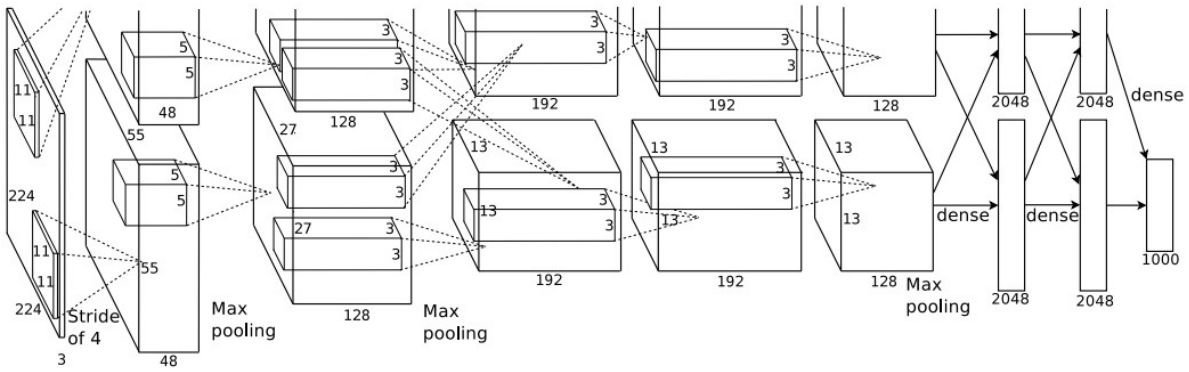


Figure 1: The Architecture of CNN in the original paper[2]

We implement the model by ourselves using keras package in python, and our model is exactly the same as the model used in the original paper as we can see in Figure 1. There are eight layers in total, and first five layers are convolutional and last three are fully-connected. The first convolutional layer filters the $224 \times 224 \times 3$ input image with 96 kernels of size $11 \times 11 \times 3$ with a stride of 4 pixels. The second convolutional layer takes as input the (response-normalized and pooled) output of the first convolutional layer and filters it with 256 kernels of size $5 \times 5 \times 48$. The third, fourth, and fifth convolutional layers are connected to one another without any intervening pooling or normalization layers. The third convolutional layer has 384 kernels of size $3 \times 3 \times 256$ connected to the (normalized, pooled) outputs of the second convolutional layer. The fourth convolutional layer has 384 kernels of size $3 \times 3 \times 192$, and the fifth convolutional layer has 256 kernels of size $3 \times 3 \times 192$. The fully-connected layers have 4096 neurons each.

What's more, as described in the original paper, we train our model using stochastic gradient descent with a batch size of 128 examples, momentum of 0.9, and weight decay of 0.0005.

3.2 Datasets

In the original paper, experments are basically conducted on ImageNet and its subsets (ILSVRC-2010/2012). ImageNet is a large visual database contains more than 14 million images which belongs to more than 20,000 categories. ILSVRC uses a subset of ImageNet with roughly 1000 images in each of 1000 categories. In all, there are roughly 1.2 million training images, 50,000 validation images, and 150,000 testing images.

The original paper also conduct experiments on relatively small datasets, such as , NORB [16], Caltech-101/256 [8, 9], and CIFAR-10/100. Due to the limitation of time and resources, we mainly conduct experiments on a subset of CIFAR-10. Cifar-10 dataset is an image dataset that has 10 classes and 60000 images, the image is stored in the form of a $32 * 32$ matrix. We use default setting from keras.datasets package as our split of training and test data, which is 50000 for training and 10000 for testing. What's more, in order to reduce the overfitting, we artificially standardize and enlarge the dataset using label-preserving transformations.

3.3 Hyperparameters

In the training process, all the hyperparameters and setups are set corresponding to the original experiment. To be specific, we do have two extra experiments to change the activation function and the dropout respectively to test the reproductivity of the paper. Those are discussed in section 4.2.1 and 4.2.2.

What's more, we want a appropriate epoch number overall. We first tried to use 100 epochs in reproducing the original experiment, and it worked and produced a reasonable and conclusive results. After that, we chose to keep 100 epochs in the following extra experiments.

3.4 Experimental setup and code

We used keras.layers library from python to implement the model. To be specific, we used Conv2D, Dense to construct each convolutional layers and fully-connected layers respectively. Also, the response-normalization and pooling after first, second and fifth convolutional layers is built by using MaxPooling2D. What's more, during the learning process, the function SGD is used to construct the stochastic gradient descent. All the hyperparameters and setups are according to the original model which is discussed in section 3.1.

You may see our code in the code.zip file.

3.5 Computational requirements

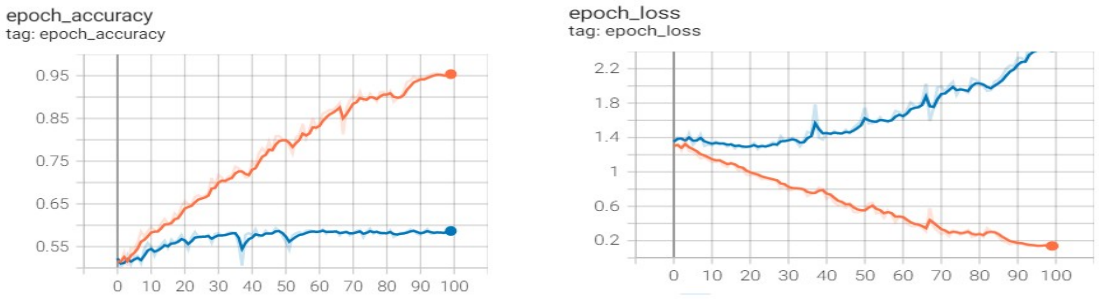
The running time of the first experiment is $03:17:28$, $03:22:33$ for the second,$03:17:25$ for the third, and $01:48:59$ for the last. All of the experiments are run with the default GPU and memory setting of Colab.(around 12GB RAM and 1 cpu)

4 Results

4.1 Results reproducing original paper

4.1.1 Result 1

This experiment tests the original model without any modification. Training accuracy continues increasing as epoch increases, which is reasonable. Validation accuracy converges in about 20 iterations, and the maximum validation accuracy is 59.33% at epoch 48. From the plots, we can tell overfitting problem gets worse after about 40 epochs.



(a) Epoch Accuracy (b) Epoch Loss

Figure 2: Original Model

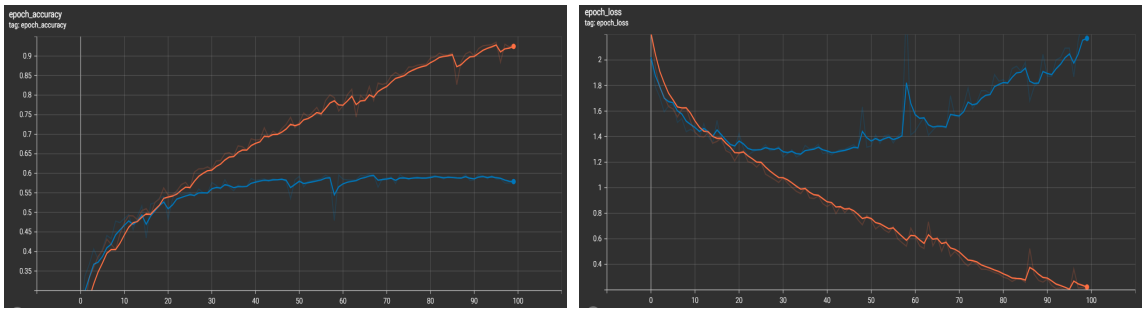
Maximum Training Accuracy	Maximum Validation Accuracy	Time Used
95.76%	59.33%	03:22:33

4.2 Results beyond original paper

We also ran our models under different settings, including reduced dropout, different activation functions, and reduced layers.

4.2.1 Additional Result 1

This experiment tests the model with reduced dropout. The original model has a dropout of 0.5, while this model has a dropout of 0.01. Validation accuracy converges in about 30 iterations, which is much slower than the original model. The maximum validation accuracy is 59.73% at epoch 92. From the plots, we can tell overfitting problem gets worse after about 60 epochs.



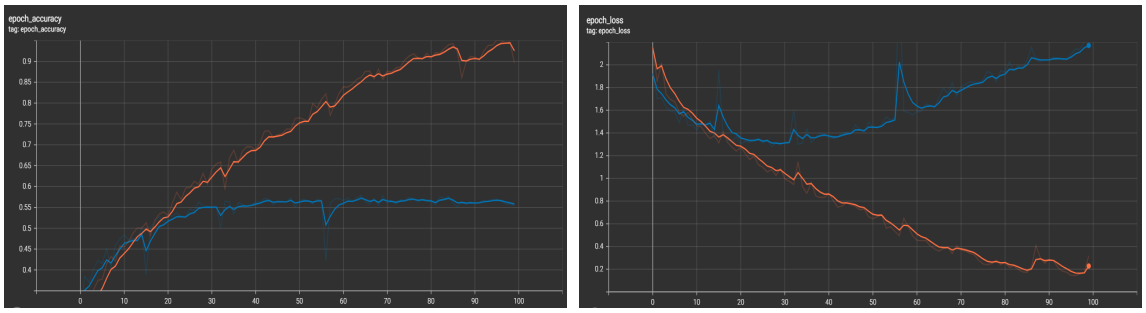
(a) Epoch Accuracy (b) Epoch Loss

Figure 3: Model with Reduced Dropout

Maximum Training Accuracy	Maximum Validation Accuracy	Time Used
92.48%	59.73%	03:17:28

4.2.2 Additional Result 2

This experiment tests the model with tanh as activation function. The original model uses ReLU as activation function. Validation accuracy converges in about 40 iterations, which is much slower than the original model. The maximum validation accuracy is 57.91% at epoch 65. From the plots, we can tell overfitting problem gets worse after about 55 epochs.



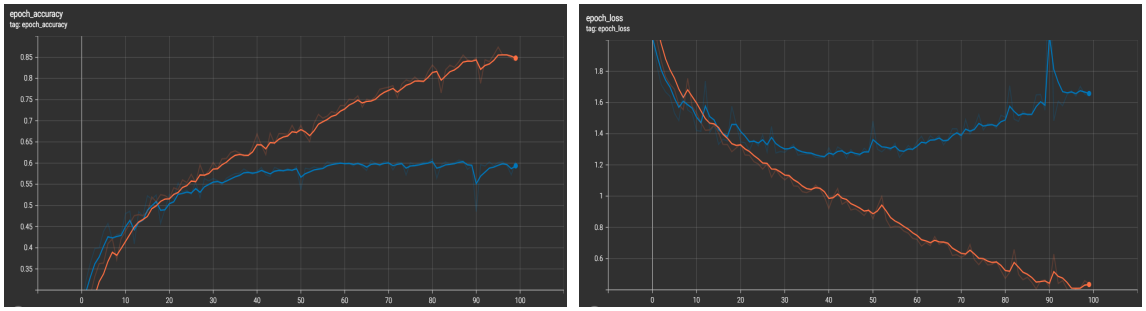
(a) Epoch Accuracy (b) Epoch Loss

Figure 4: Original Model

Maximum Training Accuracy	Maximum Validation Accuracy	Time Used
94.44%	59.71%	03:17:25

4.2.3 Additional Result 3

This experiment tests the model with four convolutional layers and three fully-connected layers. The original model has five convolutional layers and three fully-connected layers. Validation accuracy converges in about 40 iterations, which is slower than the original model. From the plots, we can tell overfitting problem gets worse after about 60 epochs. However, the maximum validation accuracy is 61.02% at epoch 81,which is higher than the maximum validation accuracy of the original model which is 59.33%.



(a) Epoch Accuracy (b) Epoch Loss

Figure 5: Original Model

Maximum Training Accuracy	Maximum Validation Accuracy	Time Used
85.62%	61.02%	01:48:59

5 Discussion

For the first experiment, the maximum validation accuracy is 59.33% after running 100 epochs, and the Top-1 error rate in the original paper is 40.9% when training with one convolutional neural network. The only difference of our experiment compared to the original one is we use CIFAR-10 dataset whereas the original experiment uses a subset of ImageNet. From here, we can conclude that our experiment fits the original paper very well, and thus we successfully reproduce the original experiment.2

Then, from our extra experiments, we can conclude that our experiments support the second; while support of the first and the third claim is not included.

To be specific, as Figure 3 and the table4.2.1 show in extra experiment 1, we can see that the maximum training accuracy is 92.48%, and the maximum validation accuracy is 59.73% after reducing the dropout rate. However, compared to

section 4.1.1, the training accuracy is reduced by 3.28%, and the validation accuracy is increased by 0.40%. The overfitting even decreases when dropout rate is reduced. This result surprised us a lot, and due to our current knowledge, we think that might beaause .

In the second extra experiment, we trained our model using tanh as the activation function instead of ReLu. From Figure 4, program run time for reaching a 25% training error is 1:40:41, achieved at 50-60 epochs; and run time of section 4.1.1 is 01:25:16, achieved at 40-50 epochs. When using tanh, program run time when reaching a 25% training error is 15 minutes less than using ReLu. Therefore we can conclude that ReLU provides faster convergence speed than tanh, and our experiment supports the second claim.

Finally, we cut one convolutional layer in our CNN in the third extra experiment. From Figure 5 and table, the maximum validation accuracy is 61.02%, and model run time is 01:48:59. Compared with section 4.1.1, the accuracy increased by 1.69%, and time used was reduced by almost 50%. After discussion, we concluded that might because of the high learning rate. Someone has enough time and resources may reduce the learning rate to 10^-6 to 10^-7 but to compensate increase the number of epochs to 10^6 to 10^7 may resolve that problem.

Testing of performance on ImageNet dataset is also not included in this paper because of technical infeasibility.

5.1 What was easy

The authors used fundamental techniques of machine learning to build their optimized models, such as using ReLU as activation function and adding dropout. As a result of that, we can easily build and train our model using functions from keras and tensorflow packages. We built the overall architecture of our CNN model without many difficulty.

5.2 What was difficult

The original paper uses ImageNet to test the performance of the model. However we found that it is impossible to train ImageNet dataset in a few hours due to the limitation of time and resources. Therefore, we have to find a replacement. CIFAR-10 dataset is a desired substitute, a dataset with the same form of image but much fewer entries and classes. CIFAR-10 dataset is used as a measure of the overall performance of the authors' models as well. Because we are applying our models on a different dataset, we have to understand every aspect and every number presented in the original paper. For example, we have to fit our model to a 10-way softmax instead of 1000 in the original paper.

References

[1] Alex Krizhevsky. "Learning Multiple Layers of Features from Tiny Images". In: 2009.

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira et al. Vol. 25. Curran Associates, Inc., 2012. URL: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.