# Image Classification with CNN

Denis Kim

***Abstract*—Image classification is one of the primary domains in deep learning. And convolutional neural networks play an important role in the improvement of image classification performance. Current work presents the CNN architecture that performs an image classification on a database with 10 classes: airplane, bird, dog, frog, horse, apple, strawberry, kiwi, lemon, and grape. Presented architecture is based on a well-known LeNet5 architecture, that originally used for hand-written character recognition problems. The model was customized for 3 channel 32x32 images. And different variations of the architecture were examined. After that, the best performing one was chosen. The final model has a pattern of a convolutional layer with a relu activation followed by a max-pooling layer. The accuracy of the model on the testing data set was 85 percent that is 232/272 correctly classified images.**

## I. Introduction

Convolutional Neural Networks played a big role in a boost in the performance of image classification. Currently, there are models that are able to perform on enormously big data sets of high-resolution images with hundreds of classes. However, the proposed work will be focused on a relatively small data set with ten classes: airplane, bird, dog, frog, horse, apple, strawberry, kiwi, lemon, and grape. A comparison of the performance of several different custom CNN architectures with different hyperparameters will be presented in this article.

## II. Related Work

There are various architectures of CNN available that are proved to perform accurately on the Image Classification problem. For example, LeNet-5 [1], is one of the first widely known and successful applications of convolutional neural networks. Originally it was aimed to solve the problem of hand-written character recognition. It uses as an input one channel 32x32 image, and feed it to two convolutional layers with average pooling, and finally has three linear layers.

Another well-known CNN architecture for image classification is AlexNet [2]. It was designed to address the ImageNet Large Scale Visual Recognition Challenge. Thus its aimed to classify high-resolution images (244x244) with even 1000 categories. Although the general architecture is quite similar to LeNet-5, AlexNet model is considerably larger.

Two years after AlexNet architecture was designed, a deeper but simpler model was introduced, called VGG network [3]. It applies several convolutional layers with Relu activation function, max pooling, fully connected layers with Relu, and softmax. But the number of layers is relatively big.

## III. Description of the Work

First of all, I started with data preparation. The data set was taken from two different sources, thus has two different image sizes. The first 5 classes (airplane, bird, dog, frog, horse) were taken from the Cifar-10 data set, where image resolutions are 32x32. And for the last 5 classes (apple, strawberry, kiwi, lemon, grape), the resolution of images was 100x100. The images from the last 5 classes were resized to 32x32. The training data set contained 4537 examples, that were divided into the training data of size 4000 and the validation data of size 537. The final evaluation was calculated on the test data of size 272. The training data was augmented by random rotation on 30 and 60 degrees, random horizontal and vertical flips.

The following main architectures were tried:

(1) Convolutional Layer - maxPool - Convolutional Layer - maxPool - Convolutional Layer - maxPool - Linear Layer - Linear Layer - Linear Layer - softmax

(2) Convolutional Layer - maxPool - Convolutional Layer - maxPool - Convolutional Layer - maxPool - Linear Layer Dropout - Linear Layer - Dropout - Linear Layer- softmax

(3) Convolutional Layer - maxPool - Convolutional Layer - maxPool - Convolutional Layer - maxPool - Linear Layer - Linear Layer - softmax

(4) Convolutional Layer - maxPool - Convolutional Layer - maxPool - Linear Layer - Linear Layer - Linear Layer- softmax

(5) Convolutional Layer - maxPool - Convolutional Layer - maxPool - Linear Layer - Linear Layer - softmax (was chosen)

(6) Convolutional Layer - avgPool - Convolutional Layer - avgPool - Linear Layer - Linear Layer

The following optimizers were tried:

(1) SGD

(2) Adam (was chosen)

The following kernel sizes were tried:

(1) 2 for maxPool (was chosen) and avgPool

(2) 2 for ConvLayer

(3) 3 for ConvLayer

(4) 5 for ConvLayer (was chosen)

The following strides were tried:

(1) 1 for ConvLayer (was chosen)

(2) 2 for maxPool (was chosen) and avgPool

The following paddings were tried:

(1) zero padding (was chosen)

(2) one padding

The following learning rates were tried:
(1) 0.01 (fixed)
(2) 0.001 (fixed)
(3) 0.0001 (fixed)
(4) decreasing from 0.001 to 0.0001
(5) decreasing from 0.001 to 0.0005 to 0.0001 (was chosen)

The following number of epochs were tried:
(1) 10 epochs
(2) 20 epochs
(3) 30 epochs (was chosen)
(4) 40 epochs

The following batch sizes were tried:
(1) 32
(2) 64 (was chosen)
(3) 128

The final model thus can be summarized in the table I and table II

The role of the ConvLayers is to reduce the images into a form that is easier to process, without losing features which are critical for getting a good prediction. First ConvLayer reduces the input 32x32 image into the 28x28 image using a kernel of size 5 and extracts the high-level features Similar to the ConvLayer, the MaxPool layer is responsible for reducing the spatial size of the convolved feature. The First MaxPool layer reduces the size of the 28x28 input to 14x14 and is responsible for extracting dominant features. The second pair of layers have a similar purpose as the first pair. After that, the output is flattened into a column vector and fed to the fully connected layer. Finally, the Softmax Classification technique is applied.

Time for training the model with 30 epochs using GPU Nvidia GTX1050 with cuda is 1 minute 23 seconds.

There are functions train() and test() that were taken from the Lab Assignment 6.

## IV. Experiments

Lets consider the performance of several architectures with different hyper parameters that were discussed in Section 3. The numbers in brackets are correspond to the numbers in brackets in Section 3.

1) Version 1: architecture (1), optimizer (2), kernel size of 1st ConvLayer (3), kernel size of 2nd and 3rd ConvLayer (3) and (1) for maxPool, strides (1) and (2), padding (1), learning rate (2), number of epochs (2), and batch size (2)

**Accuracy of Version 1: 79%**

2) Version 2: architecture (2), optimizer (2), kernel size of 1st ConvLayer (3), kernel size of 2nd and 3rd ConvLayer (3) and (1) for maxPool, strides (1) and (2), padding (1), learning rate (2), number of epochs (2), and batch size (2)

**Accuracy of Version 2: 84%**

3) Version 3: architecture (3), optimizer (2), kernel size (1) and (3), strides (1) and (2), padding (2), learning rate (2), number of epochs (2), and batch size (2)

**Accuracy of Version 3: 73%**

4) Version 4: architecture (4), optimizer (2), kernel size (1) and (4), strides (1) and (2), padding (1), learning rate (2), number of epochs (2), and batch size (2)

**Accuracy of Version 4: 76%**

5) Version 5: architecture (5), optimizer (2), kernel size (1) and (5), strides (1) and (2), padding (1), learning rate (2), number of epochs (2), and batch size (2)

**Accuracy of Version 5: 82%**

6) Version 6: architecture (6), optimizer (2), kernel size (1) and (4), strides (1) and (2), padding (1), learning rate (2), number of epochs (2), and batch size (2)

**Accuracy of Version 6: 85%**

7) Version 7: architecture (5), optimizer (1), kernel size (1) and (4), strides (1) and (2), padding (1), learning rate (2), number of epochs (2), and batch size (2)

**Accuracy of Version 7: 74%**

8) Version 8: architecture (5), optimizer (2), kernel size (1) and (4), strides (1) and (2), padding (1), learning rate (1), number of epochs (2), and batch size (2)

**Accuracy of Version 8: 66%**

9) Version 9: architecture (5), optimizer (2), kernel size (1) and (4), strides (1) and (2), padding (1), learning rate (3), number of epochs (2), and batch size (2)

**Accuracy of Version 9: 80%**

10) Version 10: architecture (5), optimizer (2), kernel size (1) and (4), strides (1) and (2), padding (1), learning rate (4), number of epochs (2), and batch size (2)

**Accuracy of Version 10: 80%**

11) Version 11: architecture (5), optimizer (2), kernel size (1) and (4), strides (1) and (2), padding (1), learning rate (5), number of epochs (3), and batch size (2)

**Accuracy of Version 11: 85%**

12) Version 12: architecture (5), optimizer (2), kernel size (1) and (4), strides (1) and (2), padding (1), learning rate (5), number of epochs (4), and batch size (2)

**Accuracy of Version 12: 83%**

13) Version 13: architecture (5), optimizer (2), kernel size (1) and (4), strides (1) and (2), padding (1), learning rate (2), number of epochs (2), and batch size (1)

**Accuracy of Version 13: 82%**

14) Version 14: architecture (5), optimizer (2), kernel size (1) and (4), strides (1) and (2), padding (1), learning rate (2), number of epochs (2), and batch size (3)

**Accuracy of Version 14: 83%**

15) Version 15: architecture (1), optimizer (2), kernel size (3) and (1), strides (1) and (2), padding (2), learning rate (2), number of epochs (2), and batch size (2)

**Accuracy of Version 15: 81%**

All the networks above can be found in the 'networks' folder.

## V. Conclusion

Fifteen different versions of CNN architecture for ten class image classification problems were presented in this article and the best solution among them was identified. The proposed model has an accuracy of 85 percent (232/272) on the test data. It can be seen that CNN can be efficiently used for

TABLE I
FINAL MODEL ARCHITECTURE

| Layer | Input Size | Output Size | Kernel Size | Stride | Padding | Input Channel | Output Channel |
|---|---|---|---|---|---|---|---|
| ConvLayer + Relu | 32 | 28 | 5 | 1 | 0 | 3 | 32 |
| MaxPool | 28 | 14 | 2 | 2 | 0 | 32 | 32 |
| ConvLayer + Relu | 14 | 10 | 5 | 1 | 0 | 32 | 64 |
| MaxPool | 10 | 5 | 2 | 2 | 0 | 64 | 64 |
| Linear | - | - | - | - | - | 1600 | 120 |
| Linear | - | - | - | - | - | 120 | 10 |

TABLE II
HYPER PARAMETERS

| | |
|---|---|
| Optimizer | Adam |
| Batch size | 64 |
| Number of epochs | 30 |
| Learning Rate | decreasing from 0.001 to 0.0005 to 0.0001 |

image classification, but the design of the network and values of hyperparameters are crucial.

## REFERENCES


[1] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., vol. 25. Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

[3] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.