Classification of Image using Convolutional Neural Network

Abhishek Kumar (s4211650) [a] Amitoj Battu (s4330617) [a]

[a] *University of Groningen*

Abstract

In this project we have explored Convolutional Neural Networks (CNNs) for the Image classification tasks. There are two goals of this project, firstly to build different Deep architecture models like CNNs, VGG etc. to classify images. Secondly, we use several metrics for the comprehensive analysis of CNN models. For this project, we used Fashion MNIST data-set to perform multi-class classification. To do this, we used following models: a 2 layer neural network, CNN with 1 convolutional layer, CNN with 2 convolutional layers, CNN with 4 convolutional layers and batch normalization, VGG like model with 9 convolutional layers and VGG like model with batch normalization.

Index Terms - Convolutional Neural Network (CNN), Image Classification, Deep Learning, Deep Neural Networks, Computer Vision.

1 Introduction

The classification and recognition of an image comes easy for humans, but not so for the computers. To do that, an entire field of study is dedicated called Computer Vision, where we study and develop methods to analyze visual data. We have come a long way with a lot of progress over last few decades in developing methods and models that could recognize and classify images/videos or even a part of it. To achieve such feats, we use deep neural network models, also known as deep learning. In this project, we have explored different deep neural network architectures and applied them to the Fashion-MNIST dataset.

With the availability of GPUs at our disposal, a large amount of visual data and development of deep learning algorithms now has the potential to train deep neural network models, which can extract features for these difficult computer vision tasks. A lot of research has been done in last few years to develop CNN based models (like Alexnet, VGG, Resnet, GoogLeNet, etc.) to classify images with reasonably good results.

2 The Data-set

The Fashion-MNIST is a dataset of Zalando's articles image – consisting of a training set of 60,000 examples and a test set of 10,000 examples. Each examples is 28X28 gray-scale image, associated with a label from 10 classes.

Definition	Target Class
T-shirt/Top	0
Trouser	1
Pullover	2
Dress	3
Coat	4
Sandal	5
Shirt	6
Sneaker	7
Bag	8
Ankle Boot	9

Table 1: Number of classes with labels

The dataset was divided into 3 parts – training set, validation set and testing set. All the different versions of different models were trained on training data first and then were evaluated using validation set. Only the best models were used to test the testing data set.

3 Convolutional Neural Networks

Convolutional neural networks are very similar to ordinary neural networks: they are made up of neurons that have learn-able weights and biases. They are so popular for image based tasks because they make explicit assumption that the inputs are images, which allows us to encode certain properties into the model architecture. These then make the forward function more efficient to implement and vastly reduce the amount of parameters in the network.

In a typical ConvNet models, we have following layers stacked together: Convolutional Layer, Pooling Layer, and Fully-Connected Layer[1]. These can be stacked with different configurations and thus giving us different models.

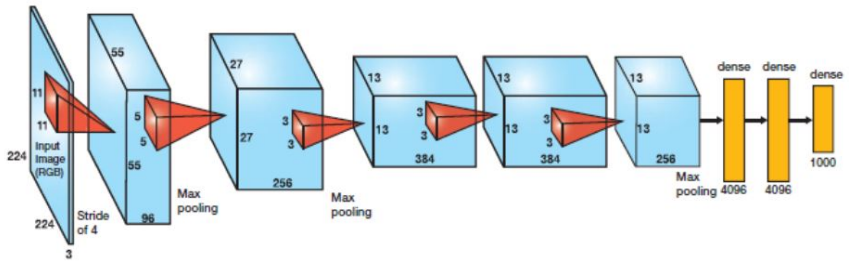


Figure 1: AlexNet, proposed by Alex Krizhevsky.

4 Image Augmentation

Deep networks need large amount of training data to achieve good performance. To build a powerful image classifier using very little training data, image augmentation is usually required to boost performance of any model[2]. Image augmentation is a technique that artificially creates training images through different ways of processing or combination of multiple processing, such as random rotation, shifts, shear and flips, etc.

We have used an augmented image generator using ImageDataGenerator API in Keras. It generates batches of image data with real-time data augmentation. After experimenting with lot of different configuration of generator, we have set our generator with following values:

Processing Tech.	Value
Rotation Range	8
Width Shift Range	0.08
Shear Range	0.3
Height shift Range	0.08
Zoom Range	0.08

Table 2: Generator Values

5 Experiments

We tried various deep neural network models with different architectures and have trained them over training data. Below are results and plots of few of our experiments.

5.1 2 layer Neural Network

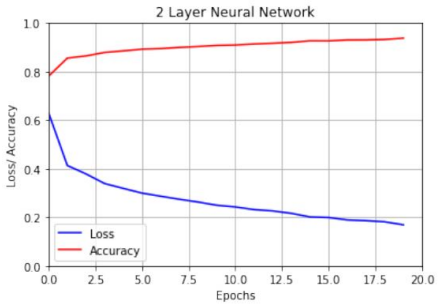
5.1.1 Hyper-parameters

Parameters	Value
Layers	2
Activation Function	Relu
Learning Rate	0.001
Total Parameters	468,474

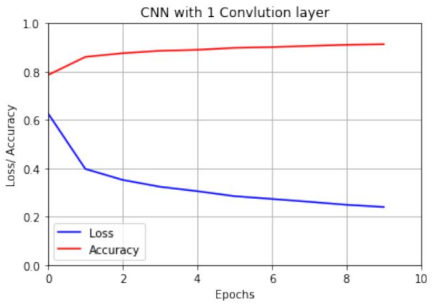
5.1.2 Evaluation Metrics

Data	Accuracy	Loss
Validation set	0.9376	0.1639
Test Set	0.8815	0.3509

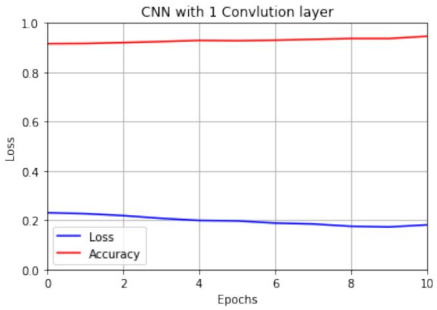
5.1.3 Plots (Loss/accuracy)



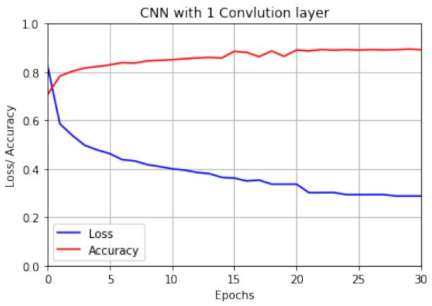
(a) with learning rate=0.001



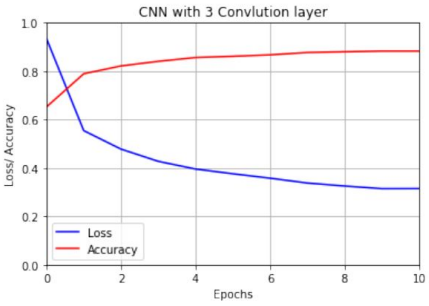
(b) with learning rate=0.0001



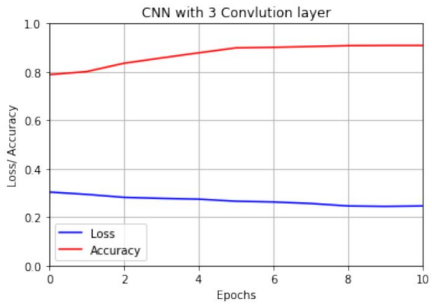
(a) with learning rate=0.001



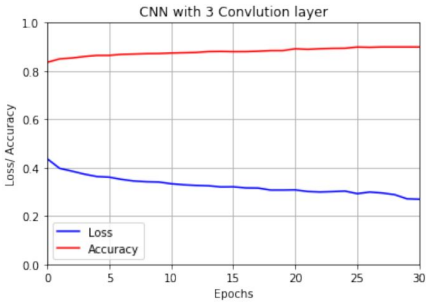
(b) with learning rate=0.0001



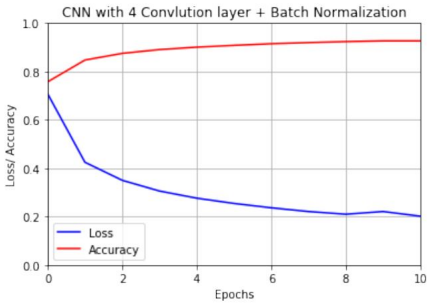
(a) with learning rate=0.001



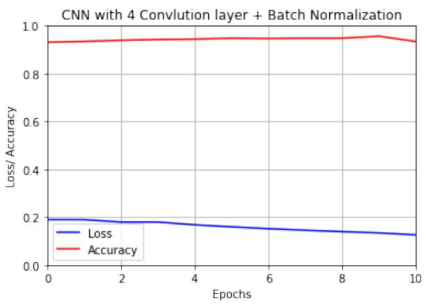
(b) with learning rate=0.0001



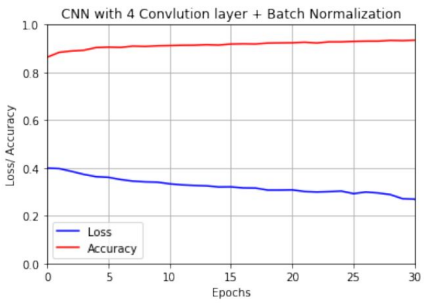
(a) with learning rate=0.001



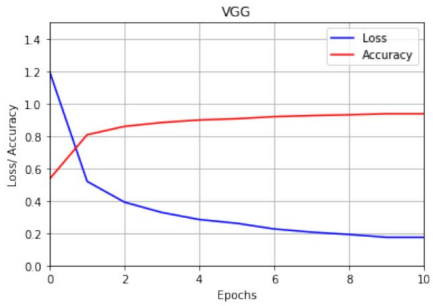
(b) with learning rate=0.0001



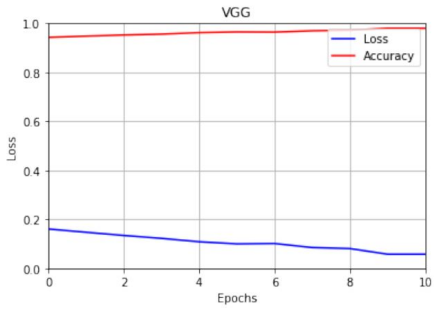
(a) with learning rate=0.001



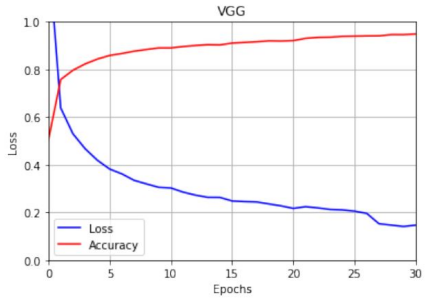
(b) with learning rate=0.0001



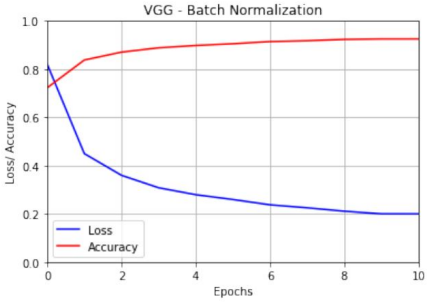
(a) with learning rate=0.001



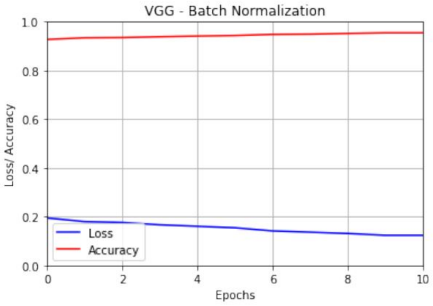
(b) with learning rate=0.0001



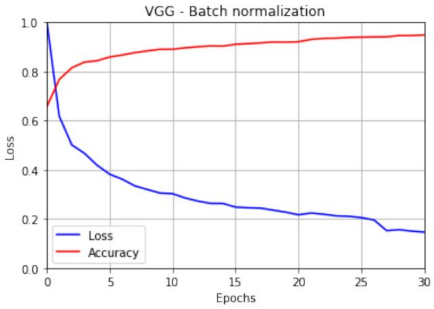
(a) with learning rate=0.001



(b) with learning rate=0.0001



(a) with learning rate=0.001



(b) with learning rate=0.0001

Figure 9: Loss/accuracy plot

5.2 Convolution Neural Network with 1 Convolutional layer:

5.2.1 Hyper-parameters:

Parameters	Value
Conv Layers	1
Pooling Layers	1
Fully Connected Layer	1
Activation Function	Relu
Learning Rate	0.001/0.0001
Total Parameters	693,962

5.2.2 Evaluation Metrics:

With learning rate=0.001,

Data	Accuracy	Loss
Validation Set	0.9132	0.2402
Test Set	0.9107	0.2533

With learning rate=0.0001,

Data	Accuracy	Loss
Validation Set	0.9376	0.1639
Test Set	0.8815	0.3509

5.2.3 Data Augmentation:

Data	Accuracy	Loss
Validation Set	0.8944	0.2879
Test Set	0.8910	0.2589

5.3 Convolution Neural Network with 3 Convolutional layer:

5.3.1 Hyper-parameters:

Parameters	Value
Conv Layers	3
Pooling Layers	2
Fully Connected Layer	1
Activation Function	Relu
Learning Rate	0.001/0.0001
Total Parameters	241,546

5.3.2 Evaluation Metrics:

With learning rate=0.001,

Data	Accuracy	Loss
Validation Set	0.8832	0.3147
Test Set	0.8815	0.3509

With learning rate=0.0001,

Data	Accuracy	Loss
Validation Set	0.9091	0.2444
Test Set	0.8935	0.2567

5.3.3 Data Augmentation:

Data	Accuracy	Loss
Validation Set	0.9130	0.2673
Test Set	0.9123	0.2360

5.4 Convolution Neural Network with 4 Convolutional layer and Batch Normalization:

5.4.1 Hyper-parameters:

Parameters	Value
Conv Layers	4
Pooling Layers	1
Fully Connected Layer	1
Activation Function	Relu
Learning Rate	0.001/0.0001
Total Parameters	241,546

5.4.2 Evaluation Metrics:

With learning rate=0.001,

Data	Accuracy	Loss
Validation Set	0.9266	0.2019
Test Set	0.9223	0.2171

With learning rate=0.0001,

Data	Accuracy	Loss
Validation Set	0.9563	0.1171
Test Set	0.9303	0.2200

5.4.3 Data Augmentation:

Data	Accuracy	Loss
Validation Set	0.9361	0.18805
Test Set	0.9272	0.2312

5.5 VGG Model:

5.5.1 Hyper-parameters:

Parameters	Value
Conv Layers	9
Pooling Layers	4
Fully Connected Layer	1
Activation Function	Relu
Learning Rate	0.001/0.0001
Total Parameters	241,546

5.5.2 Evaluation Metrics:

With learning rate=0.001,

Data	Accuracy	Loss
Validation Set	0.9376	0.1639
Test Set	0.8815	0.3509

With learning rate=0.0001,

Data	Accuracy	Loss
Validation Set	0.9217	0.3178
Test Set	0.9185	0.3376

5.5.3 Data Augmentation:

Data	Accuracy	Loss
Validation Set	0.9498	0.1413
Test Set	0.9268	0.2298

5.6 VGG with batch normalization:

5.6.1 Hyper-parameters:

Parameters	Value
Conv Layers	9
Pooling Layers	4
Fully Connected Layer	1
Activation Function	Relu
Learning Rate	0.001/0.0001
Total Parameters	241,546

5.6.2 Evaluation Metrics:

With learning rate=0.001,

Data	Accuracy	Loss
Validation Set	0.9153	0.2003
Test Set	0.9253	0.2001

With learning rate=0.0001,

Data	Accuracy	Loss
Validation Set	0.9325	0.1240
Test Set	0.9311	0.1195

5.6.3 Data Augmentation:

Data	Accuracy	Loss
Validation Set	0.9492	0.1404
Test Set	0.9371	0.1863

6 Observations

6.1 Accuracy

Out of all models, VGG model with batch normalization performed the best with almost 95% accuracy.

Model	Accuracy(%)
2 Layer Neural Network	88
CNN with 1 Conv layer	91
CNN with 3 Conv layer	91
CNN with 4 Conv layer + batch normalization	92
VGG model	93
VGG model + batch normalization	95

6.2 Loss

Model	Accuracy(%)
2 Layer Neural Network	88
CNN with 1 Conv layer	91
CNN with 3 Conv layer	91
CNN with 4 Conv layer + batch normalization	92
VGG model	93
VGG model + batch normalization	95

6.3 Depth of Model and Image Augmentation

As we increase depth of our model by increasing the number of layers we get much more accuracy and our loss decreases. It was also observed that with augmented images, we got slightly better accuracy.

7 Conclusion

Among all experiments, the VGG model performed the best with almost 95% accuracy. We conclude that CNN based models performed really well on images and their performance can be increased or improved with carefully adding more convolutional layers. Processing images before putting them into models can also have significant impact on the model performance.

References

[1] Alex Krizhevsky. Imagenet classification with deep convolutional neural networks. *NIPS 2012*, 2012.

[2] Karen Simonyan. Very deep convolutional networks for large-scale image recognition. *ICLR 2015*, 2015.