Image Classification using Convolutional Neural Networks

Pradeep Singh
pradeepsingh7890@live.com
Computational Science Research Center, San Diego State University
CA, USA

***Abstract* — This project explores Convolutional neural network (CNNs) for Image classification tasks. I have used state-of-the-art Convolutional neural networks (CNN) based models to classify images. The goal of this project is twofold: 1) To build different neural network (CNNs, VGG etc.) models that can classify given images. 2) Comprehensive analysis of CNN models using various metrics. In this project I perform multi-class classification over Fashion MNIST dataset using following models: a simple 2 layer neural network, CNN with 1 convolutional layer, CNN with 2 convolutional layer, CNN with 4 convolutional layers and batch normalization, VGG like model with 9 convolutional layers and VGG like model with batch normalization.**

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

I. Introduction

The classification and recognition of an image is an easy task for humans, but for computers it's not easy at all. An entire field of study/ research is dedicated to it called – computer vision, where scientists and engineers study and develop methods to analyze visual data. Over the last few decades, a lot of progress has been made in developing models and methods that could classify/ recognize images/ videos or even parts of it. And, a larger part of this progress has come from deep neural networks models, also known as deep learning. This project explores different deep neural network models and architectures and apply them over Fashion MNSIT dataset.

With advent of GPUs, availability of large amount of visual data and development of deep learning algorithms brought the potential to train deep neural network models to extract features for these challenging computer vision tasks. In last few years, a lot of CNN base models (Alexnet, VGG, ResNet, GoogLeNet, etc.) has been developed that can successfully classify images with state of the art results.

II. Data set

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

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

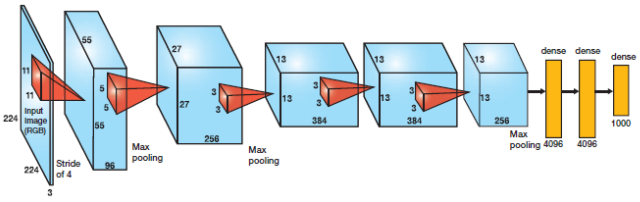
Fig 1: Number of classes with labels

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.

III. Convolutional Neural Networks

Convolutional neural networks are very similar to ordinary neural networks: they are made up of neurons that have learnable 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. These can be stacked with different configurations and thus giving us different models.



AlexNet, proposed by Alex Krizhevsky.

IV. 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. 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.

I 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, I have set my 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

V. Experiments

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

A. 2 Layer Neural Network

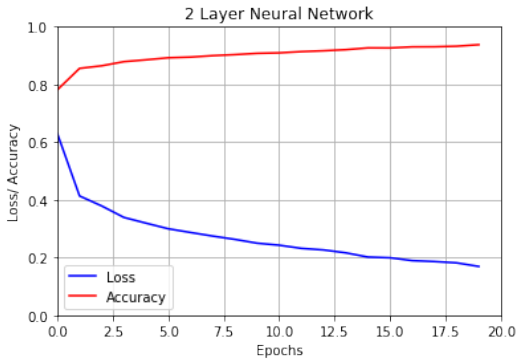
I. Hyper-parameters

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

II. Evaluation Metrics

Data	Accuracy	Loss
Validation Set	0.9376	0.1639
Test Set	0.8815	0.3509

III. Plots



B. Convolutional Neural Network with 1 Convolution layer:

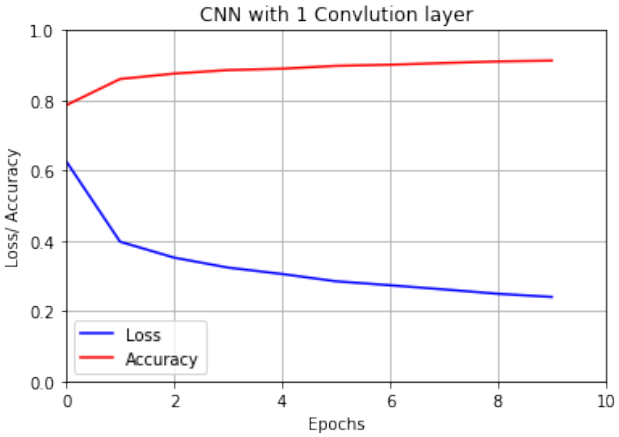
I. Hyper-parameters:

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

II. Evaluation Metrics:

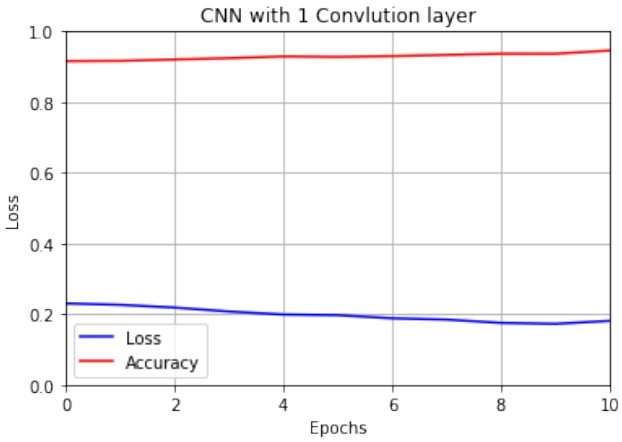
With learning rate = 0.001,

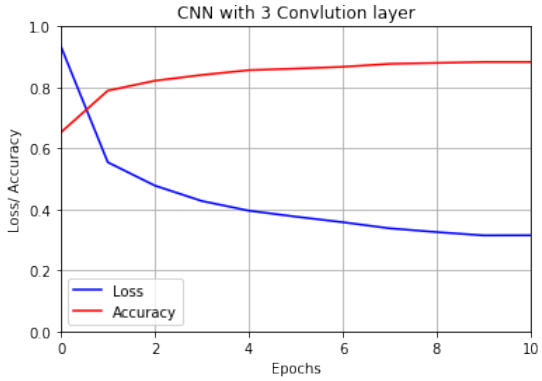
Data	Accuracy	Loss
Validation Set	0.9132	0.2402
Test Set	0.9107	0.2533



With learning rate = 0.0001

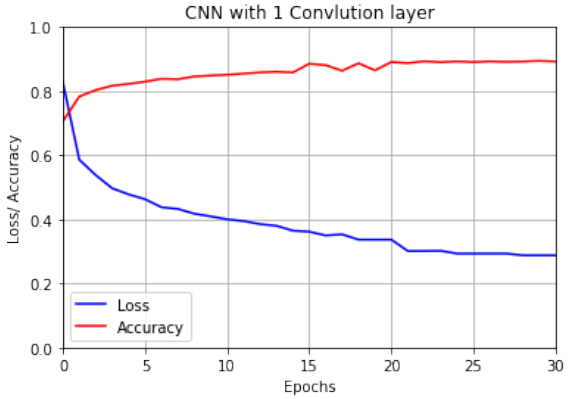
Data	Accuracy	Loss
Validation Set	0.9376	0.1639
Test Set	0.8815	0.3509





III. Data Augmentation:

Data	Accuracy	Loss
Validation Set	0.8944	0.2879
Test Set	0.8910	0.2589



C. Convolutional Neural Network with 3 Convolution layer:

I. 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

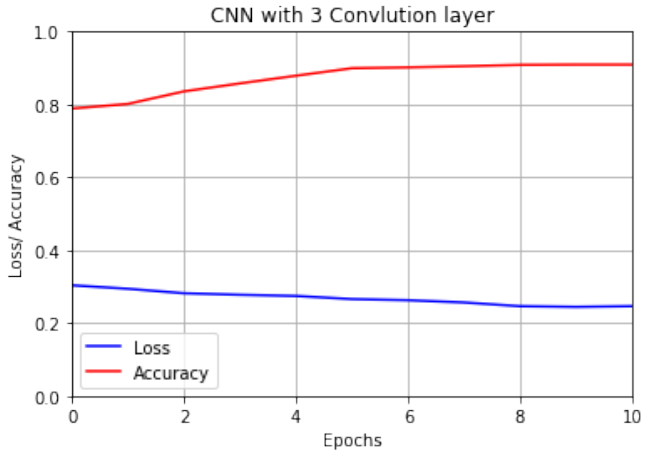
II. Evaluation Metrics

With learning rate = 0.001,

Data	Accuracy	Loss
Validation Set	0.8832	0.3147
Test Set	0.8815	0.3509

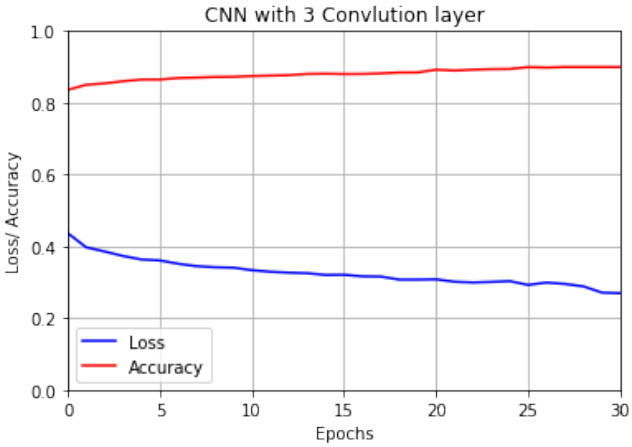
With learning rate = 0.0001,

Data	Accuracy	Loss
Validation Set	0.9091	0.2444
Test Set	0.8935	0.2567



III. Data Augmentation

Data	Accuracy	Loss
Validation Set	0.9130	0.2673
Test Set	0.9123	0.2360



D. Convolutional Neural Network with 4 Convolution layer and Batch Normalization

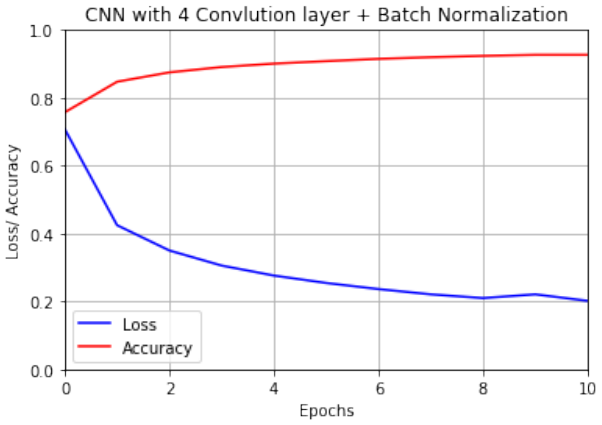
I. Hyper-parameters

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

II. Evaluation Metrics

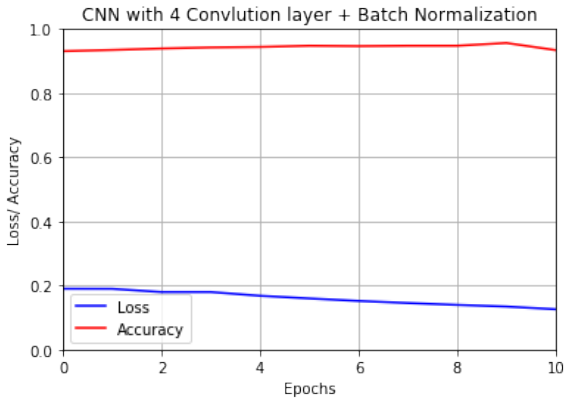
With learning rate = 0.001,

Data	Accuracy	Loss
Validation Set	0.9266	0.2019
Test Set	0.9223	0.2171



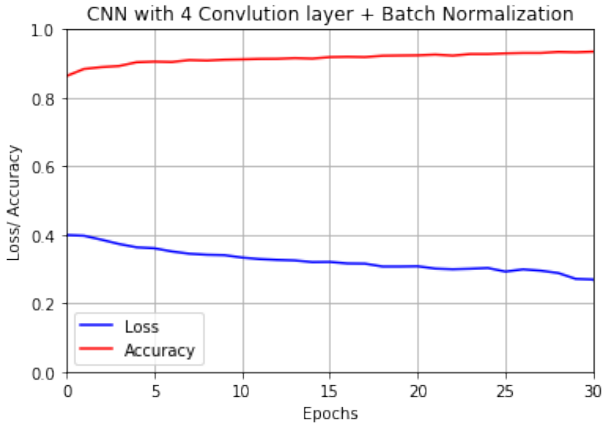
With learning rate = 0.0001,

Data	Accuracy	Loss
Validation Set	0.9563	0.1171
Test Set	0.9303	0.2200



III. Data Augmentation

Data	Accuracy	Loss
Validation Set	0.9361	0.18805
Test Set	0.9272	0.2312



E. VGG Model

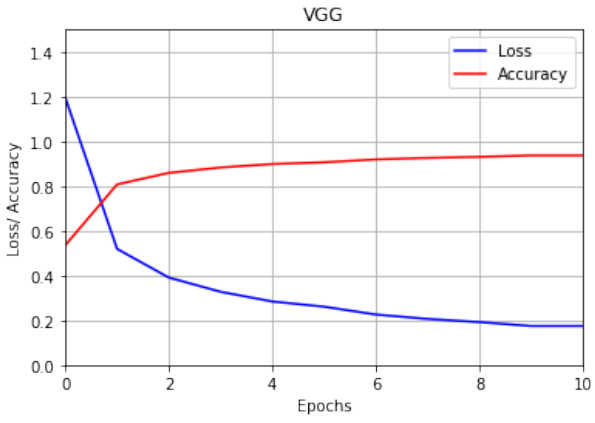
I. Hyper-parameters

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

II. Evaluation Metrics

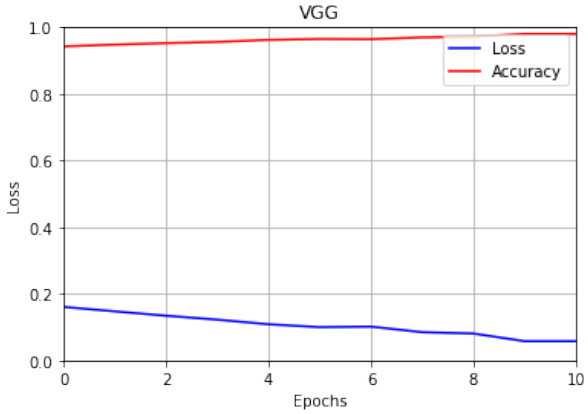
With learning rate = 0.001,

Data	Accuracy	Loss
Validation Set	0.9376	0.1639
Test Set	0.8815	0.3509



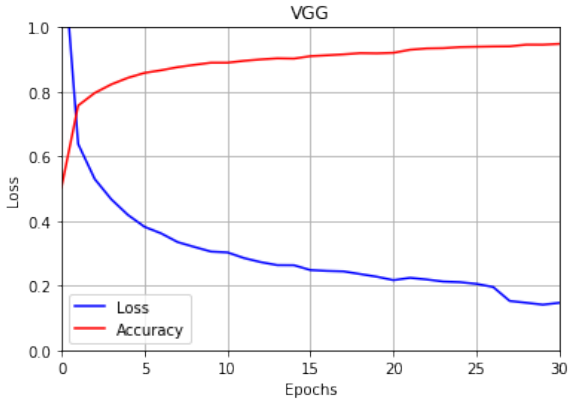
With learning rate = 0.0001,

Data	Accuracy	Loss
Validation Set	0.9217	0.3178
Test Set	0.9185	0.3376



III. Data Augmentation

Data	Accuracy	Loss
Validation Set	0.9498	0.1413
Test Set	0.9268	0.2298



F. VGG with batch normalization:

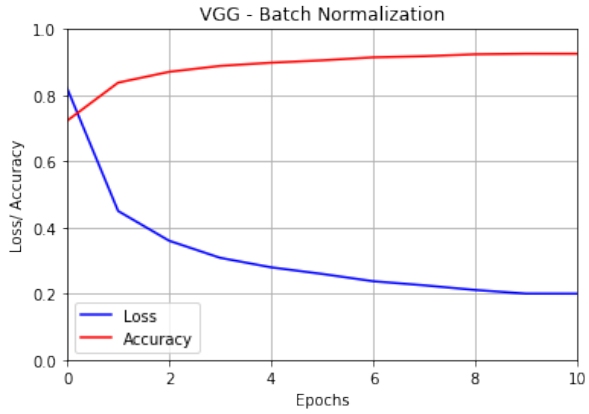
I. Hyper-parameters

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

II. Evaluation Metrics

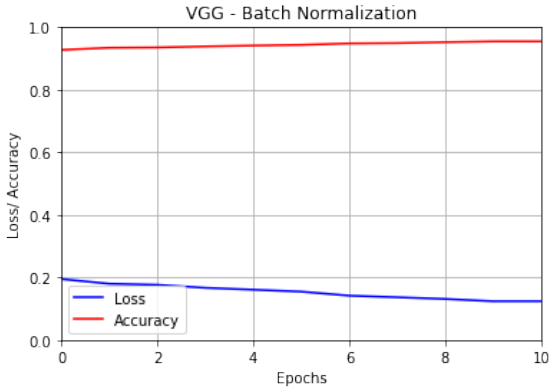
With learning rate = 0.001,

Data	Accuracy	Loss
Validation Set	0.9153	0.2003
Test Set	0.9253	0.2001



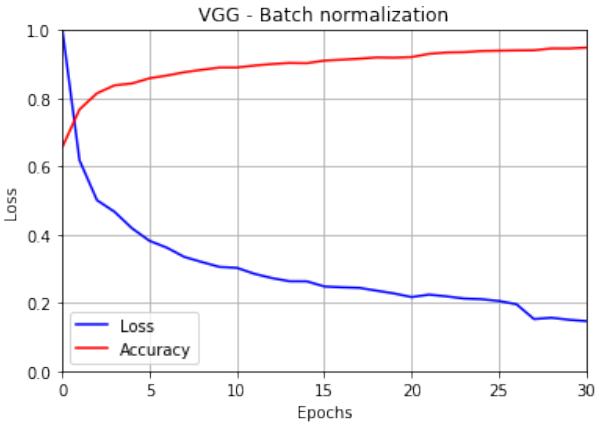
With learning rate = 0.0001,

Data	Accuracy	Loss
Validation Set	0.9325	0.1240
Test Set	0.9311	0.1195



III. Data Augmentation

Data	Accuracy	Loss
Validation Set	0.9492	0.1404
Test Set	0.9371	0.1863



VI. Observations

A. 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 4 Conv layer + batch normalization	92
VGG model	93
VGG model + batch normalization	95

B. Loss

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

C. 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.

VII. Conclusion

Among all experiments, model VGG 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 layers. Processing images before putting them into models can also have significant impact on model performance.

References

[1] Alex Krizhevskv (2012, Dec). ImageNet Classification with Deep Convolutional Neural Networks. Presented at NIPS 2012. [Online]. Available: https://papers.nips.cc

[2] Karen Simonyan (2015, March). Very Deep Convolutional Networks for large-scale Image recognition. Presented at ICLR 2015. [Online]. Available: https://iclr.cc/

[3] Kaming He. (2016, June). Deep Residual Learning for Image Recognition. Presented at IEEE CVPR 2016. [Online]. https://arxiv.org/pdf/1512.03385.pdf

[4] Christian Szegedy. (2015, Sep). Going Deeper with Convolutions. Presented at CVPR 2015. [Online]. https://ieeexplore.ieee.org/document/7298594/

[5] Jason Wang. The Effectiveness of Data Augmentation in Image Classification using Deep Learning. [Online]. http://cs231n.stanford.edu/reports/2017/pdfs/300.pdf

[6] Fei-Fei Li, Justin Johnson, Serena Yeung. Course: CS231n : Convolutional Neural Networks for Visual Recognition. http://cs231n.stanford.edu/

[7] Image Augmentation for Deep Learning with Keras. https://machinelearningmastery.com/image-augmentation-deep-learning-keras/

[8] Fashion-MNIST Dataset. https://github.com/zalandoresearch/fashion-mnist

Author:

I (Pradeep Singh) did this project as part of CS – 596 Machine Learning course in Spring 2018 at SDSU. I did everything from project brainstorming, to exploring datasets, diving data in 3 sets, writing models in TensorFlow and Keras and conducting experiments with different model architectures, training different models over Google cloud, analyzing results and finally evaluating trained models over testing data. I later reported all my results by writing this report.

I would also like to thank Dr. Xiaobai Liu for providing guidance and feedback from time to time basis.