Deep Learning
First Practical Project

Sudhakaran Jain (S3558487) [a] Subilal Vattimunda Purayil (S3587630) [a]

[a] *University of Groningen, P.O. Box 72 9700 AB Groningen*

Abstract

This project presents a comparative study of performances of various Convolutional Neural Networks(CNN) by changing the parameter settings. All these CNNs were trained on 'Fashion-MNIST' [3] dataset for image classification. We then compare their performances with respect to various optimizers, activation functions, data augmentation and regularization techniques used. Finally, we take the best performing CNN among others and compare it with our model of ResNet [1] and VGGNet [2].

1 Introduction

The aim of the project was to compare and discuss the effect of changing parametric settings of Activation Functions, Optimizers, Regularization techniques and Data Augmentation on the performance of Convolutional Neural Networks(CNN). These different CNNs which were trained on an image dataset for classification are then compared to see which setting performs better for every given parameter. The performance is measured based on two aspects namely accuracy of the CNN on unknown data and number of epochs it took during training. Finally, we will also see how the best CNN thus obtained performs in comparison with other well known CNNs namely ResNet and VGGNet.

2 The Dataset

To train all the different CNNs, we use 'Fashion-MNIST' dataset which is a widely used dataset. It is a well known replacement for original MNIST dataset for benchmarking machine learning algorithms. It consists of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28x1 grayscale image, associated with a label from 10 classes.

3 The Experimental Setup

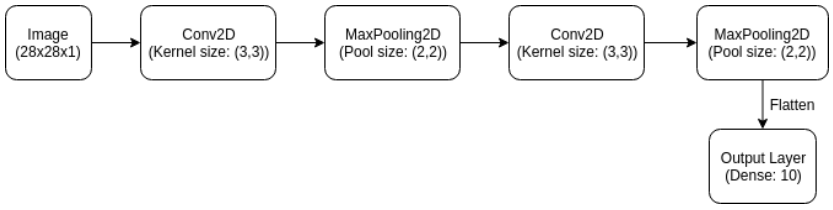

Figure 1: Network Architecture of CNNs present in setting-1 to setting-7

We trained 7 CNNs of different settings and our model of ResNet and VGGNet on Fashion-MNIST dataset. Those 7 CNNs have the same base architecture as shown in Figure 1. The implementation was done using Keras[1] framework. The Table 1 shows the different parameter settings of each CNN along

Table 1: Parameter settings of all CNNs

Optimizer	ID	Activation Function	Regularization	Data Augmentation	Accuracy	Epochs
SGD with momentum: 0.9	Setting-1	ReLU	None	No	88.76%	30
	Setting-2	ReLU	Dropout,Batch Normalization,Weight Decay	No	90.80%	100
	Setting-3	Sigmoid	Dropout,Batch Normalization,Weight Decay	No	90.15%	150
	Setting-4	PReLU	Dropout,Batch Normalization,Weight Decay	No	91.46%	200
RMSProp with ρ: 0.9	Setting-5	Sigmoid	Dropout,Batch Normalization,Weight Decay	No	90.36%	50
Adam with $\beta 1$: 0.9 and $\beta 2$: 0.999	Setting-6	Sigmoid	Dropout,Batch Normalization,Weight Decay	No	90.63%	100
	Setting-7	Sigmoid	Dropout,Batch Normalization,Weight Decay	Yes	89.15%	200
	Setting-8	ResNet-v2			86.37%	110
	Setting-9	VGGNet			93.37%	30

with their performances. All the CNNs were trained in a minibatch of 170, with same learning rate as 0.001. We also plotted accuracy-vs-epoch graph for each CNN to examine further.

4 Results and Observations

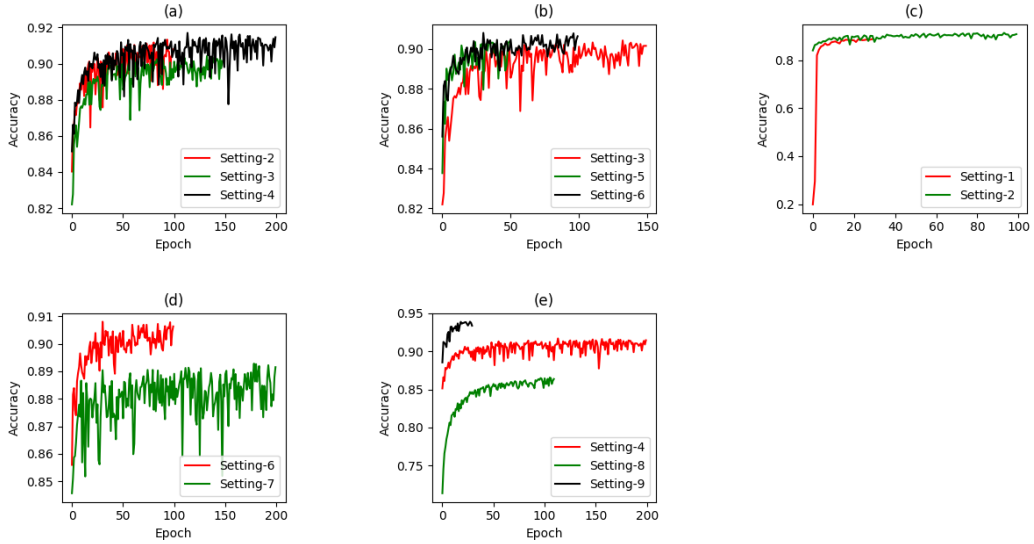

Figure 2: Performance graphs for each comparison

We intended to compare the effect of different parameters settings namely activation functions, optimiz-

ers, regularization techniques, data augmentation on the performance of CNN. Hence, while comparing different settings for a parameter, we kept rest of the parameters as same to observe true differences. We picked the best CNN among others based on accuracy.

4.1 Comparison of different Activation Functions

For this, we considered setting-2, setting-3 and setting-4. Here, we obtained the highest accuracy in case of PReLU compared to other activation functions. Moreover, from Figure 2(a) we observe that the graph for PReLU started at slightly higher accuracy rate from the first epoch itself, while Sigmoid started from the lowest. But, PReLU also took more number of epochs to train. ReLU on the other hand, took least number of epochs while also achieving a good accuracy rate.

4.2 Comparison of different Optimizers

Here, we compared SGD with momentum, RMSProp and Adam optimizer by considering setting-3, setting-5 and setting-6. The highest accuracy was obtained in case of Adam optimizer as we can see clearly in Figure 2(b). The Adam optimizer has both momentum as well as RMSProp parameters terms in it. This explains why it outperformed other optimizers.

4.3 Comparison based on usage of Regularization Techniques

For this, we compared setting-1 which did not have any regularization technique with setting-2 which had Dropout, Batch Normalization as well as Weight Decay. We observed that the accuracy of setting-2 was better than setting-1, though it took more number of epochs to be trained. Moreover, Figure 2(c) shows that setting-2 attained much higher accuracy starting from the first epoch itself. One more interesting observation that we noted was that the CNN of setting-1 sometimes got stuck in local optima during training resulting in a very low accuracy rate. This problem is solved in setting-2 by the use of regularization.

4.4 Comparison based on usage of Data Augmentation

To see the effect of data augmentation, we considered setting-6 and setting-7. To our surprise, the CNN in setting-6 which was trained without any data augmentation acquired a bit more accuracy compared to CNN of setting-7 which was trained on augmented data. We feel this may be due the smaller size of the image which is just 28x28x1. Even a small change in terms of rotation or a shift changes the aspects of the image to a great extent. Due to this the CNN of setting-7 was not able to extract sufficient features during training. Moreover, from Figure 2(d) it is evident that setting-7 took much more epochs to train as the dataset size increased due to augmentation.

4.5 Comparison of best CNN with ResNet and VGGNet

From Table 1, we observe that the best CNN (having highest accuracy of 91.46%) is obtained in setting-4. We compared this with our ResNet and VGGNet models. We then observed that VGGNet obtained the highest accuracy of 93.37% among rest of them as seen in Figure 2(e). Surprisingly, it also took much lesser number of epochs to be trained. Though, we observed that each of these epochs took a bit more time to execute compared to the epochs in other two CNNs. The fact that VGGNet has many deep layers compared to other CNNs explains this observation.

References

[1] François Chollet et al. Keras. https://keras.io, 2015.

[2] Adrian Rosebrock. pyimagesearch. https://www.pyimagesearch.com/2019/02/11/fashion-mnist-with-keras-and-deep-learning/, 2019.

[3] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.