CS7015 : DEEP LEARNING

PROGRAMMING ASSIGNMENT 3

- CONVOLUTIONAL NEURAL NETWORKS -

March 30, 2018

Student ID:
Namida M - EE15B123
Ganga Meghanath - EE15B025

Contents

1 INTRODUCTION

The Fashion-MNIST dataset contains images of 10 classes of clothing apparel. The training data contains around 55000 images.

2 DATA AUGMENTATION

We dealt with 4 different sets of training files, including the one provided. The augmentation was carried out separately for each class and the datapoints were sampled from the same to generate new training data.

2.1 Train 1

The raw train set provided was used without any augmentation and the results were observed. The test accuracy was found to be around 89% using the same.

2.2 Train 2

The augmentations carried out has been tabulated below.

flip left right	probability=0.5
random zoom	probability=0.5
rotate($-5^o, 5^o$)	probability=0.5
flip top bottom	probability=0.005

10,000 datapoints were sampled from each of the augmented class dataset and was appended to the original train dataset and the resultant train matrix was shuffled and utilised for training. Hence, we had over 1,55,000 datapoints in the training set. This was employed for the best model as it gave the best results as compared to the other augmented training sets.

2.3 Train 3

The augmentations carried out has been tabulated below.

flip left right	probability=1
random zoom	probability=0.5
rotate(-5^o,5^o)	probability=1
random erasing	probability=0.5

2.4 Train 4

The augmentations carried out has been tabulated below.

flip left right	probability=1
random zoom	probability=0.5
rotate(-5^o,5^o)	probability=1
random erasing	probability=0.5

3 BEST MODEL

Configuration details :

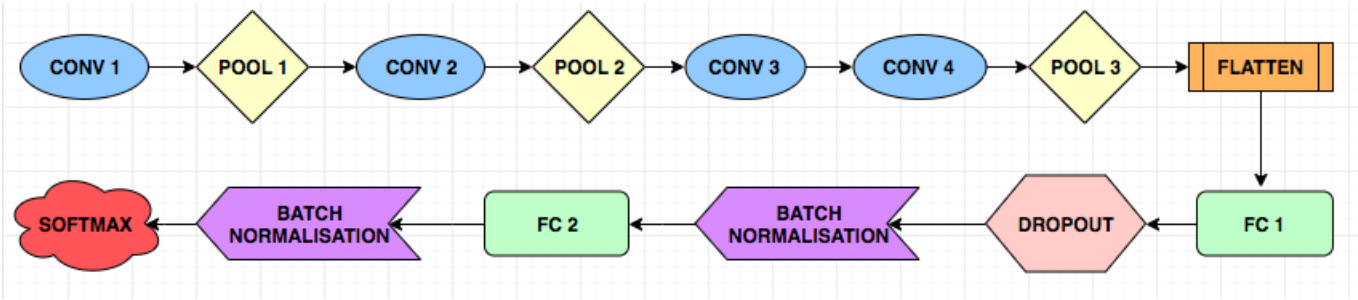
Layer	Input Shape	Output Shape	Kernel size	Parameters	No. of neurons
CONV1	(None,28,28,1)	(1,28,28,64)	(3,3)	640	50176
POOL1	(None,28,28,64)	(1,14,14,64)	(2,2)	0	12544
CONV2	(None,14,14,64)	(1,14,14,128)	(3,3)	73856	25088
POOL2	(None,14,14,128)	(1,7,7,128)	(2,2)	0	6272
CONV3	(None,7,7,128)	(1,7,7,256)	(3,3)	295168	12544
CONV4	(None,7,7,256)	(1,7,7,256)	(3,3)	590080	12544
POOL3	(None,7,7,256)	(1,4,4,256)	(2,2)	0	4096
FLATTEN	(1,4,4,256)	(None, 4096)	—	0	4096
FC	(None, 4096)	(None, 256)	—	1048832	256
FC	(None, 256)	(None, 256)	—	65792	256
SOFTMAX	(None, 256)	(None, 10)	—	2570	10

Note : The no.of neurons are calculated as $height \times width \times depth$
Parameters include both weights and biases.

The no of parameters are calculated as,

$$filter\_size \times filter\_size \times input\_depth \times output\_depth + output\_depth$$

3.1 Architecture



Relu activation function was used in all layers except for the final layer (softmax). The keep probability of dropout was set to 0.4.

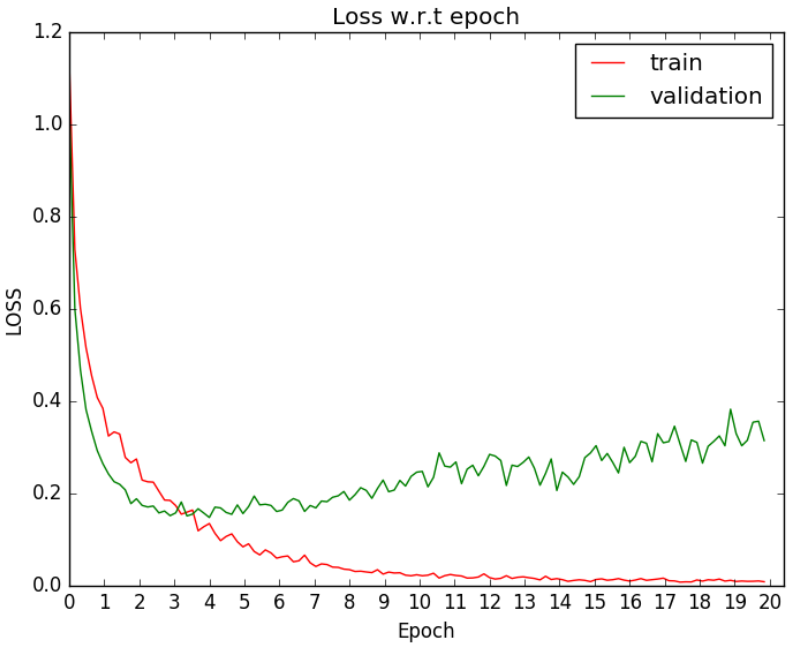
3.2 Training details

The hyperparameters used during training are tabulated below. The code was trained first on our system and later on Google Colab.

Hyperparameters	
Learning Rate	0.001
Optimiser	Adam
Batch Size	250
Initialiser	Xavier
Activation	Relu
Epochs	16
Patience	5

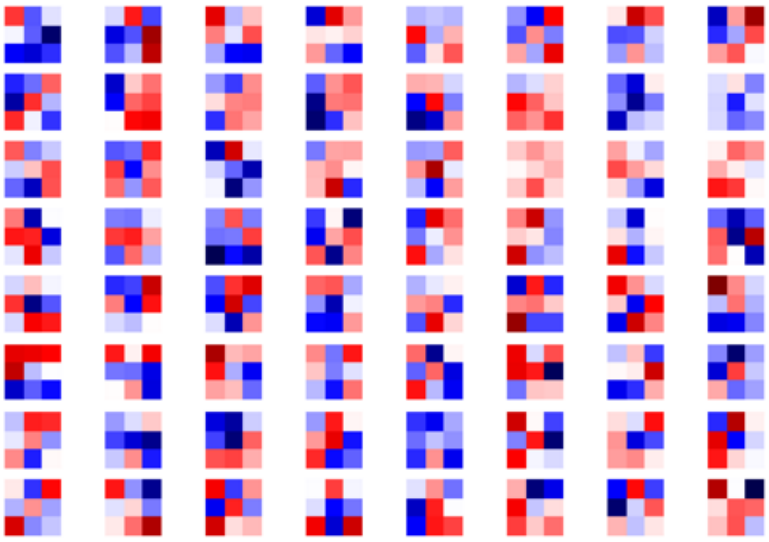
3.3 PLOTS

3.3.1 Learning Curves



3.3.2 CONV1 filter weights

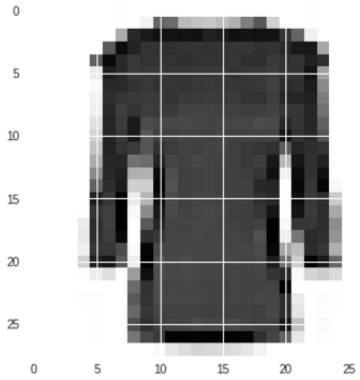
The learned weights of the 64 filters in the first convolution layer has been plotted below. The negative weights have been given in blue and the positive weights have been given as red. The intensity of the color is a measure of the magnitude of the weights.



3.4 Output of Convolution layers

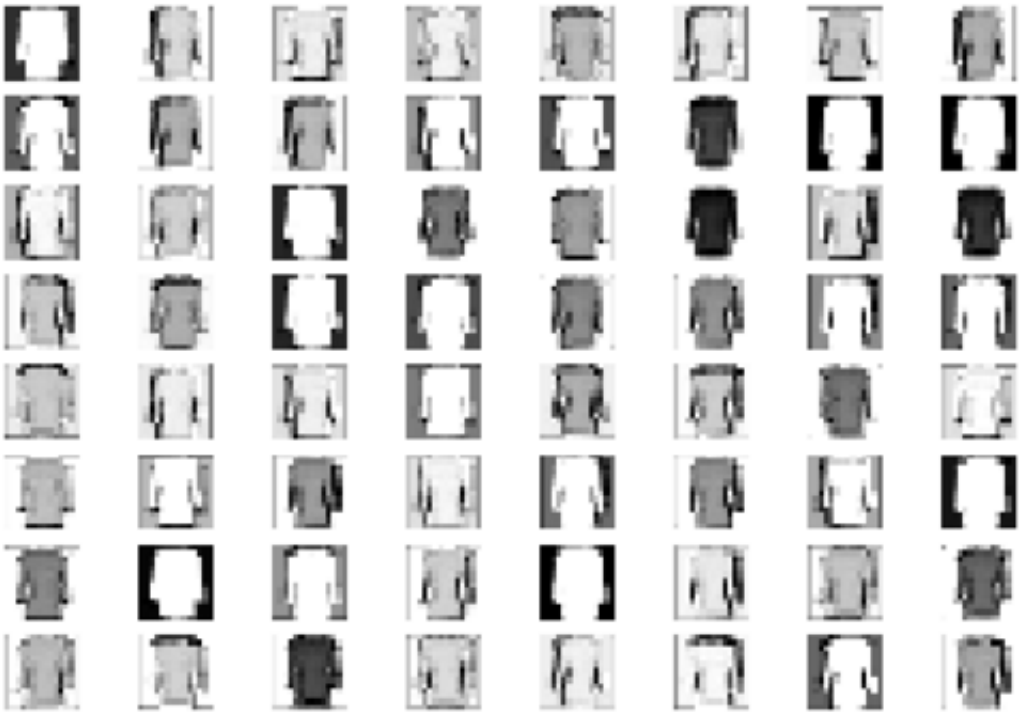
The output of the filters of the convolution layers after feeding in a test image have been shown below. This allows us to study what each filter learns and analyze the trends starting from the input image to the last convolution layer.

As the layer progress, we notice the features extracted by each layer are more abstract compared to those of the previous layer. We pass the following input into the network:



The activations of each layer are as follows:

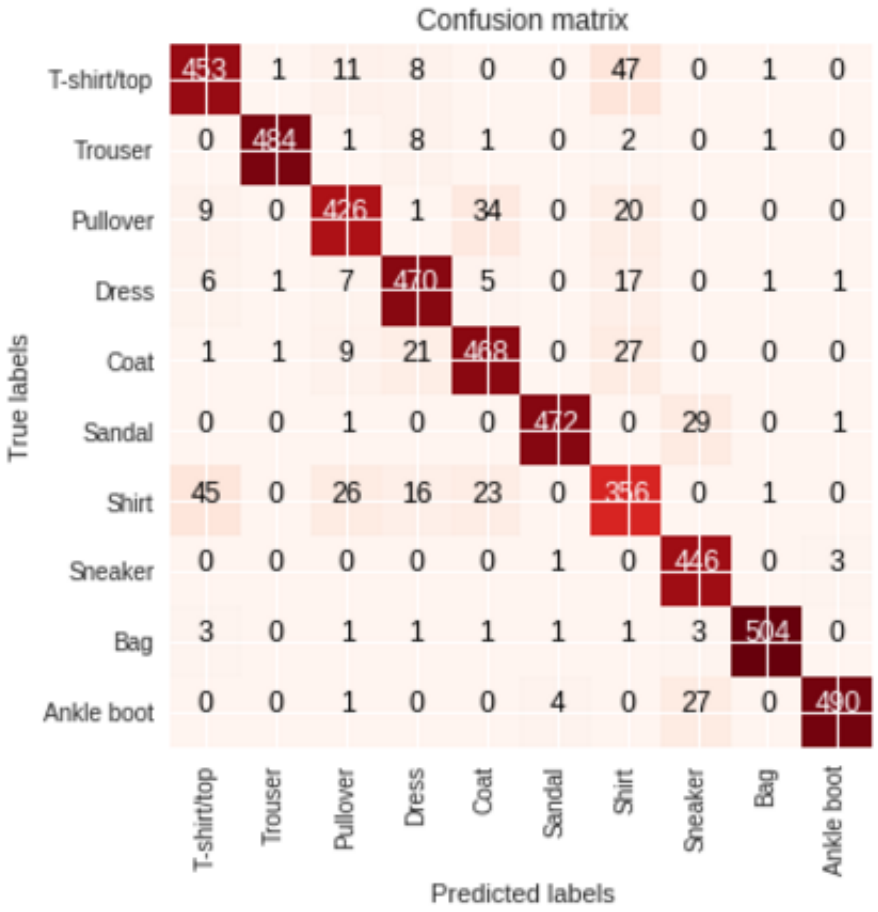
3.4.1 CONV 1



3.4.2 CONV 2

3.4.3 CONV 3

3.4.4 CONFUSION MATRIX



3.5 Network specs

As we can observe from the table given above, the one which describes the network, there are more number of parameters in the fully connected layers as compared to the convolution layers. In the considered architecture, the skewness in the distribution of parameters is relatively less.

The total contribution of parameters by the convolutional layers is : 959744

The total contribution of parameters by the dense layers is : 1117194

$$\begin{aligned}\text{Ratio} &= \frac{\text{No. of parameters in convolution layers}}{\text{No. of parameters in FC layers}} \\ &= \frac{959744}{1117194} \\ &= 0.859\end{aligned}$$

3.6 Effect of batch norm

To test the effect of batch normalization, the same network was considered with and without batch normalizations and was executed for 2 epochs. The dataset considered is the augmented version consisting of 1,55,000 datapoints and 1000 iterations were run for a batch size of 250. The learning rate was set as 0.001 for Adam optimiser and the activation function used was relu. The architecture followed was a slight modification of the given architecture.

Batch Normalization	**Training Accuracy**	**Validation Accuracy**
0 BN : None	74.8	83.04
1 BN : Before Softmax layer	90.4	89.16
2 BN : Before Softmax and output	85.2	89.12
3 BN : Before Softmax, output and FC1	78.8	87.22
4 BN : Before Softmax, output, FC1, FC 2	84.0	88.92

It can be noticed from the table that the performance improves upon adding batch normalization to the architecture.

3.7 Fooling the network

patterns
plot of accuracy v/s number of pixels changed on the test set.

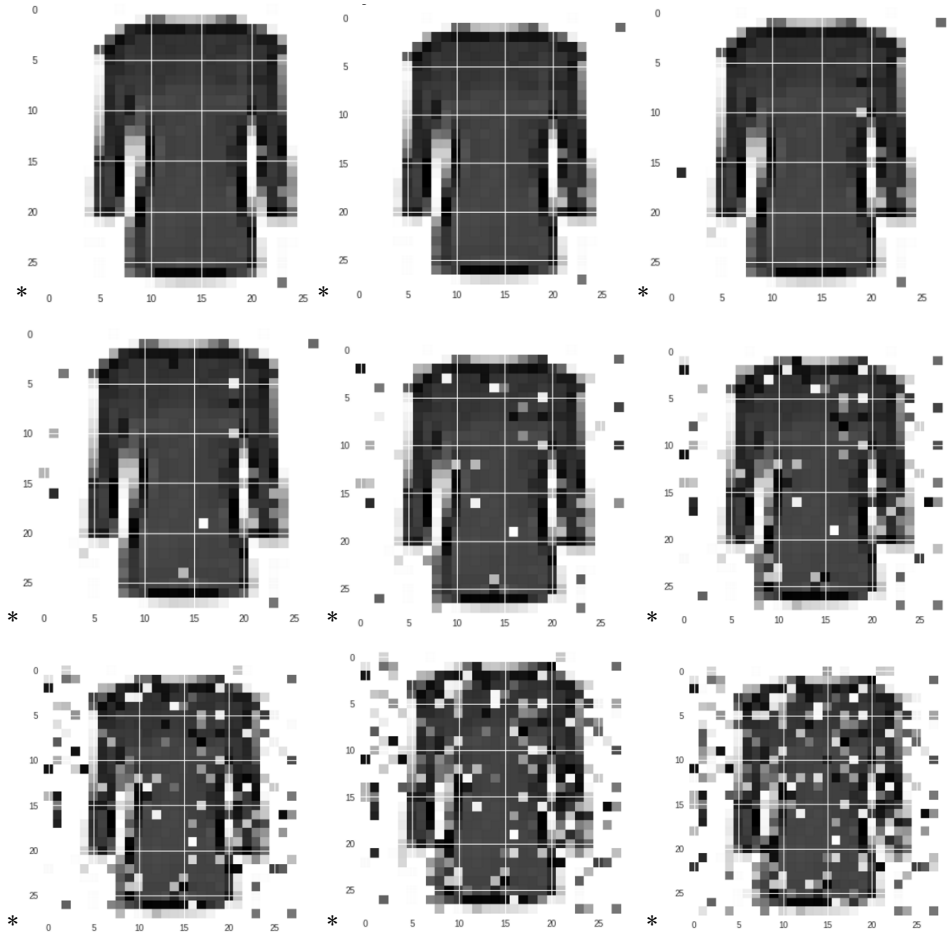
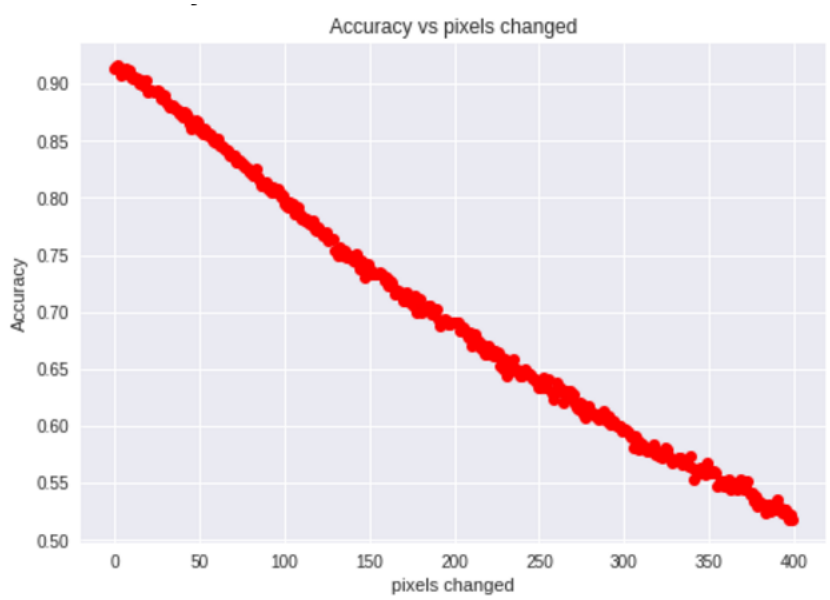


Figure 1: Images with varying levels of noise

[H]


Accuracy vs pixels changed
Accuracy
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0
50
100
150
200
250
300
350
400
pixels changed

4 APPENDIX

The following include some of the hyperparameter tuning that were carried out during the course of the assignment.

Learning Rate	10^{-4} to 0.1
Optimisation	Adam, RMSProp A, RMSProp B
Batch Size	500, 1000
FC size	256, 512, 1024
Non Linearity	Relu, Elu
Initializer	Xavier, He

The filtre sizes chosen varied between 7,5,3 and we found that 3 gave better accuracy and later sticked to filters of size 3×3.

RMSProp A	decay = 0.9
RMSProp B	decay = 0.75, momentum = 0.01

Note : For RMSProp, selected parameters giving above 45% accuracy on validation have been shown. For Adam, values above 65% accuracy on validation have been shown. Although minor tests were run using Adagrad and Momentum based gradient descent, since Adam gave better results, further models were trained using Adam. (only Adam and RMSProp has been tabulated)

1000 iterations were run for each set of parameters with a batch size of 250.(Note : Augmented dataset)

We also noticed that there were random behaviour with certain set of hyperparameters and such behaviours were found to be less for Rmsprop in comparison to adam

Learning rate	batch_size	fc_1size	non_linearity	optimiser	train acc	val acc
0.001	500	256	relu	Adam	74.58	71.6
0.01	500	256	relu	Adam	63.67	66.98
0.001	1000	256	relu	Adam	70.8	75.94
0.001	500	512	relu	Adam	67.8	74.04
0.01	500	512	relu	Adam	74.8	69.4
0.001	1000	512	relu	Adam	71.8	77.48
0.001	500	1024	relu	Adam	74.0	77.72
0.01	500	1024	relu	Adam	70.8	73.84
0.001	1000	1024	relu	Adam	70.1	74.8
0.001	500	256	relu	Adam	74.58	71.6
0.001	500	256	elu	Adam	72.2	78.58
0.001	1000	256	elu	Adam	71.6	77.64
0.001	500	512	elu	Adam	77.2	79.56
0.001	500	1024	elu	Adam	76.4	79.44
0.001	1000	1024	elu	Adam	54.9	63.99
0.01	1000	512	relu	Adam	65.3	70.89
0.0001	500	256	relu	Adam	66.8	68.99
0.0001	500	512	relu	Adam	69.8	73.82
0.0001	1000	512	relu	Adam	60.3	66.22
0.0001	500	1024	relu	Adam	69.6	74.92
0.0001	1000	1024	relu	Adam	63.4	67.16
0.0001	500	256	elu	Adam	65.6	74.26
0.0001	1000	256	elu	Adam	62.4	69.28
0.0001	500	512	elu	Adam	70.2	76.3
0.0001	1000	512	elu	Adam	66.7	73.44
0.0001	500	1024	elu	Adam	63.6	72.42
0.0001	1000	1024	elu	Adam	66.5	70.92

Learning rate	batch_size	fc_1size	non_linearity	optimiser	train acc	val acc
0.001	500	256	relu	Rmsprop_A	55.4	57.94
0.001	1000	256	relu	Rmsprop_A	41.8	46.32
0.001	500	512	relu	Rmsprop_A	58.4	63.73
0.001	500	512	relu	Rmsprop_A	48.2	52.78
0.001	1000	512	relu	Rmsprop_A	54.9	61.32
0.001	500	1024	relu	Rmsprop_A	68.0	69.52
0.01	500	1024	relu	Rmsprop_A	48.3	55.6
0.001	1000	1024	relu	Rmsprop_A	46.5	42.93
0.01	1000	1024	relu	Rmsprop_A	46.1	54.93
0.001	500	256	elu	Rmsprop_A	65.2	70.86
0.01	500	256	elu	Rmsprop_A	59.4	68.87
0.001	1000	256	elu	Rmsprop_A	53.8	59.94
0.01	1000	256	elu	Rmsprop_A	58.7	64.96
0.001	500	512	elu	Rmsprop_A	64.4	68.58
0.01	500	512	elu	Rmsprop_A	61.8	64.78
0.001	1000	512	elu	Rmsprop_A	58.7	66.52
0.001	500	1024	elu	Rmsprop_A	62.2	67.76
0.01	500	1024	elu	Rmsprop_A	60.8	67.54
0.001	1000	1024	elu	Rmsprop_A	37.5	57.62
0.01	1000	1024	elu	Rmsprop_A	56.7	63.4
0.0001	500	256	relu	Rmsprop_A	60	65.26
0.0001	500	1024	relu	Rmsprop_A	63	68.4
0.0001	500	256	elu	Rmsprop_A	74.2	77.1
0.0001	1000	256	elu	Rmsprop_A	63.8	69.98
0.0001	500	512	elu	Rmsprop_A	63.8	70.2
0.0001	500	1024	elu	Rmsprop_A	64.8	70.94
0.0001	1000	1024	elu	Rmsprop_A	60	65.82

Learning rate	batch_size	fc_1size	non_linearity	optimiser	train acc	val acc
0.001	500	256	relu	Rmsprop_B	74.6	77.79
0.001	500	512	relu	Rmsprop_B	48	56.14
0.001	500	1024	relu	Rmsprop_B	71.8	79.2
0.001	500	256	elu	Rmsprop_B	56.8	61.91
0.001	1000	256	elu	Rmsprop_B	41.9	47.52
0.001	500	512	elu	Rmsprop_B	47.2	54.88
0.001	1000	512	elu	Rmsprop_B	40.2	46.58
0.001	1000	1024	elu	Rmsprop_B	46.5	42.93
0.001	500	1024	elu	Rmsprop_B	46.4	48.26
0.001	1000	1024	elu	Rmsprop_B	44	47.36
0.01	1000	1024	elu	Rmsprop_B	47.4	47.92
0.0001	500	256	relu	Rmsprop_B	67.6	76.8
0.0001	1000	256	relu	Rmsprop_B	66.8	71.74
0.0001	500	512	relu	Rmsprop_B	71.6	75.96
0.0001	1000	512	relu	Rmsprop_B	70.2	77.89
0.0001	500	1024	relu	Rmsprop_B	75.2	79.32
0.0001	1000	1024	relu	Rmsprop_B	71.6	75.84
0.0001	500	256	elu	Rmsprop_B	69.8	77.98
0.0001	1000	256	elu	Rmsprop_B	72.9	77.54
0.0001	500	512	elu	Rmsprop_B	73.2	77.12
0.0001	1000	512	elu	Rmsprop_B	61.1	71.1
0.0001	500	1024	elu	Rmsprop_B	73.6	77.88
0.0001	1000	1024	elu	Rmsprop_B	55.6	65.36

Bibliography

[1] Mitesh M Khapra. *CS7015 Deep Learning: Lecture 10,* Indian Institute of Technology Madras, 2018