Assignment #00100

Amirreza Bagherzadeh
PennStateUniversity
PSUID: 946204639

Abstract

The assignment's main focus is to go over the Multi Layer Preceptron Neural Network and examining the effect of training on other tasks after it was trained.

1 Multi Layer Preceptron

In this part the network of 2,3,4 hidden layer was trained on 10 tasks and the accuracy over each task was calculated and the R matrix for each size of hidden layer was calculated and from that, the BWT and ACC are as follows:

1.1 Negative Log Livelihood loss function

1.1.1 Gradient Decent Optimizer

Number of Hidden Layer	ACC	BWT
2	0.35	-0.50
3	0.14	-0.47
3 dropout	0.10	-0.004
4	0.09	-0.03
4 dropout	0.10	-0.001

1.1.2 RMSProp Optimizer

Number of Hidden Layer	ACC	BWT
2	0.48	-0.44
3	0.37	-0.57
3 dropout	0.12	-0.06
4	0.33	-0.60
4 dropout	0.10	0.004

1.1.3 ADAM Optimizer

Number of Hidden Layer	ACC	BWT
2	0.48	-0.45
3	0.43	-0.50
3 dropout	0.15	-0.04
4	0.36	-0.52
4 dropout	0.11	0.0035

1.2 L1

1.2.1 2 Layer Neural Net

Optimizer	ACC	BWT
Adam	0.32	0.27
RMSProb	0.26	0.22
SGD	0.34	0.29

1.3 L2

1.3.1 2 Layer Neural Net

Optimizer	ACC	BWT
Adam	0.43	0.001
RMSProb	0.26	0.22
SGD	0.17	0.15

1.4 L1$_L2$

1.4.1 2 Layer Neural Net

Optimizer	ACC	BWT
Adam	0.32	0.29
RMSProb	0.26	0.22
SGD	0.17	0.15

1.5 Effect of DropOut

By applying dropout to the model we see that not only it wouldn't help the forgetting, it prevents the model to learn the model itself. This maybe due to the number of epochs we train our model. By going over 100 or even 150 epoch we may have better results

1.6 Effect of depth

What I expected from increasing the number of layer was that since the model could fit more non-linear model, it would be more robust to forgetting due to it's extra flexibility. However, the forgetting increases as we increase the number of layers. This also maybe due to the number of epochs and by increasing the number of epochs we may get more robust model to forgetting by increasing the layers.

1.7 Effect of optimizer

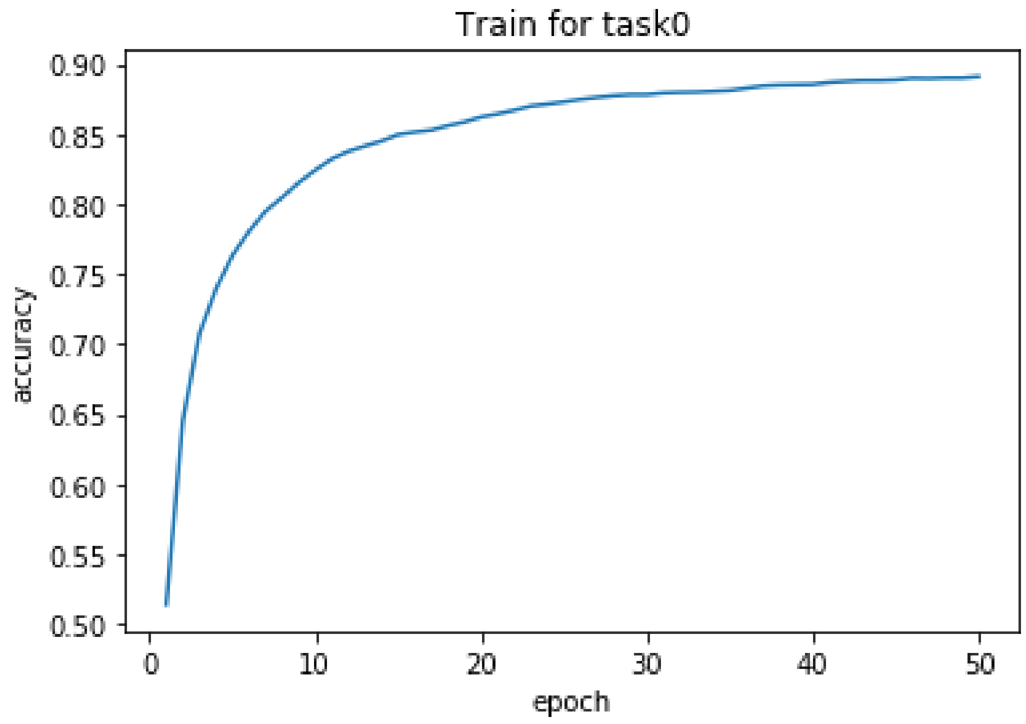
As it was seen in 1.1 the optimizer can be effective on forgetting but not by a large margin. the effect can be due to the speed that we travel from one optimum point of a task to one for the other task. Because in total, all the optimizers move to the local optimum and the only diference between them is the speed of converging.

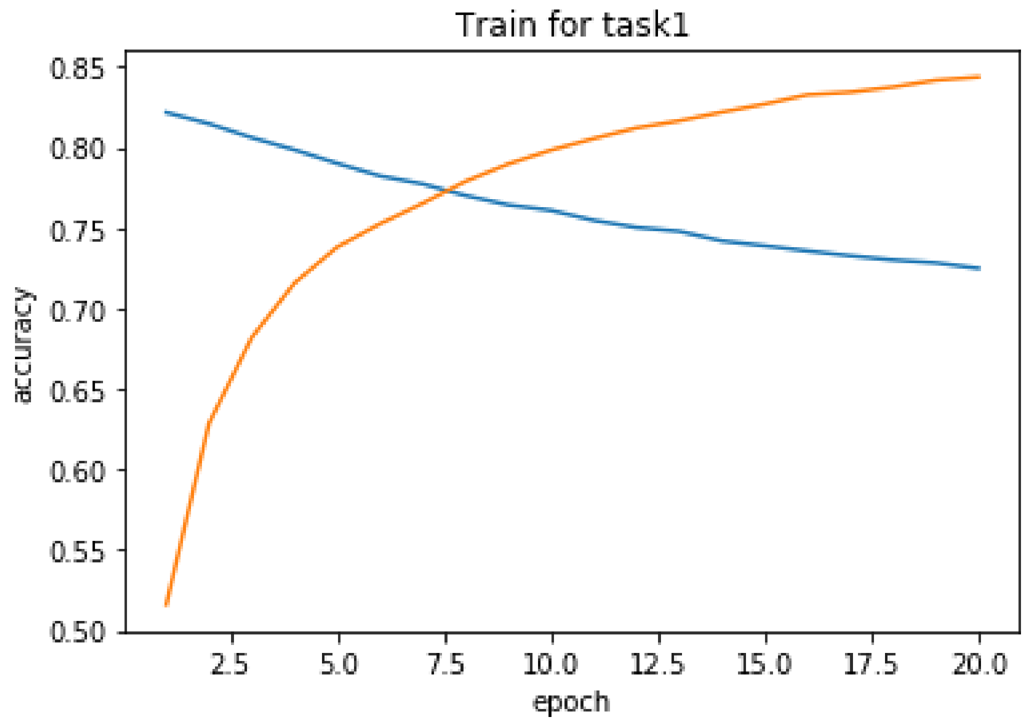
1.8 Tricks from previous assignment

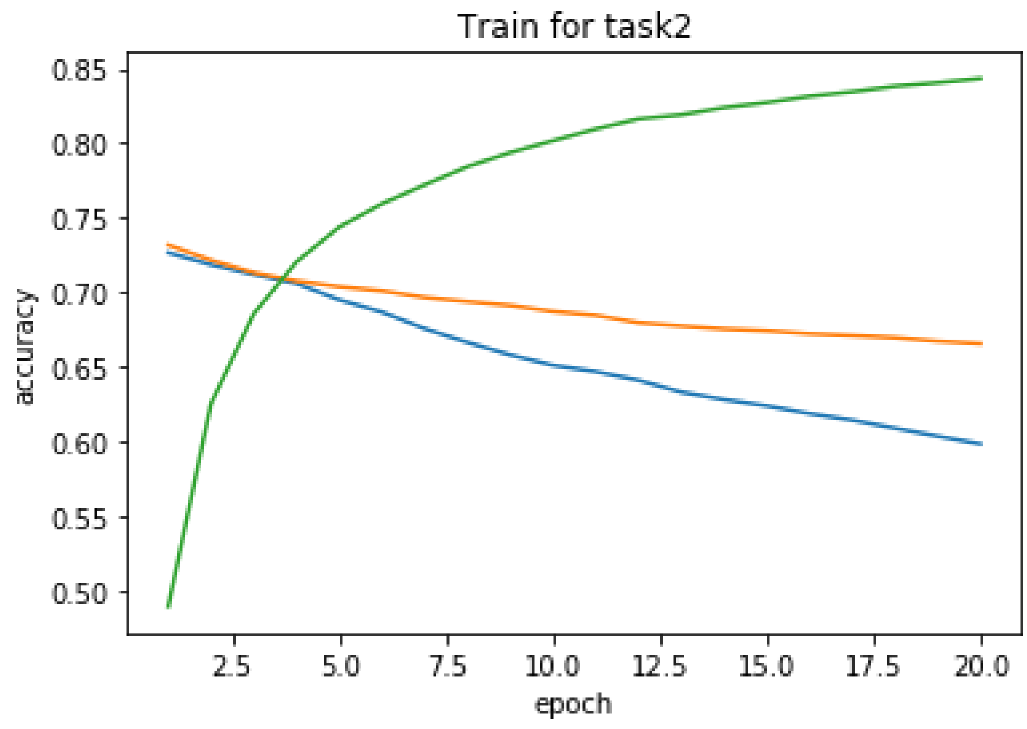
One trick we used in previous assignment was adaptive learning rate. This was not helpful in this assignment since we never got to a point were the changes from one epoch to another be so small that the change in learning be necessary.

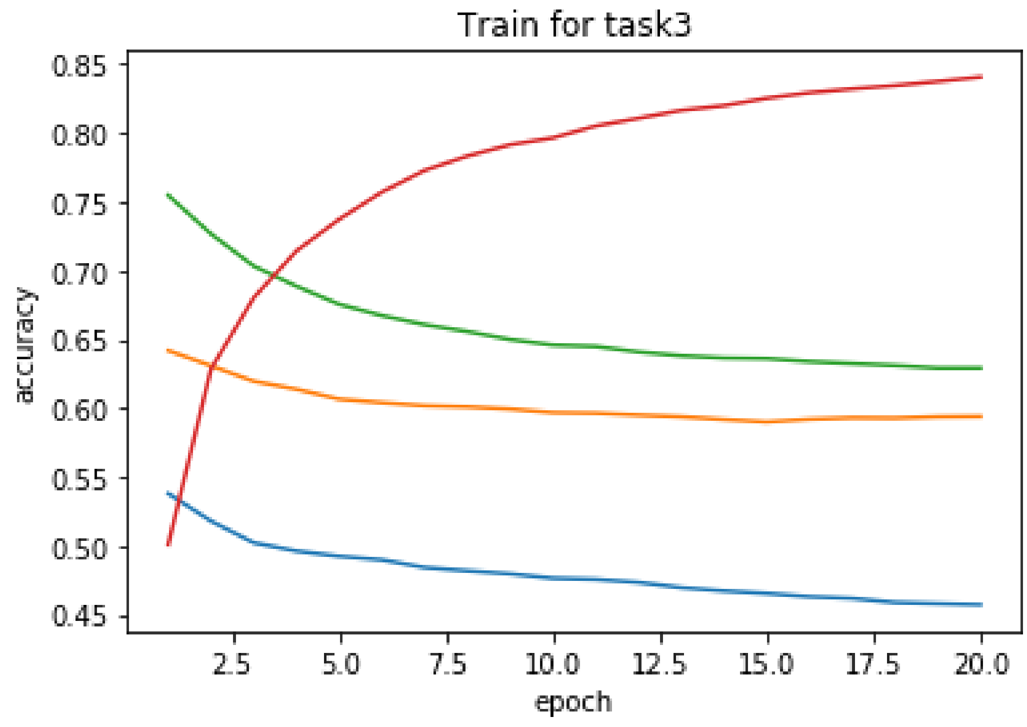
1.9 Validation Results

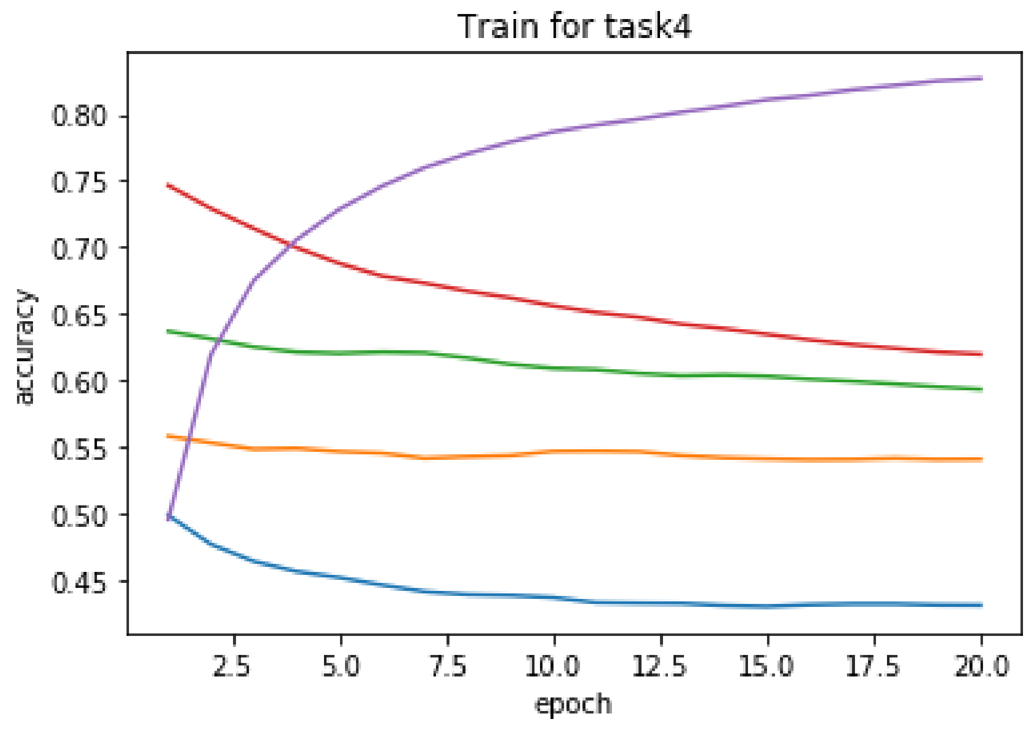
In this part Validation Results for decrease in model prediction were plotted when we have finished training on all given 10 tasks

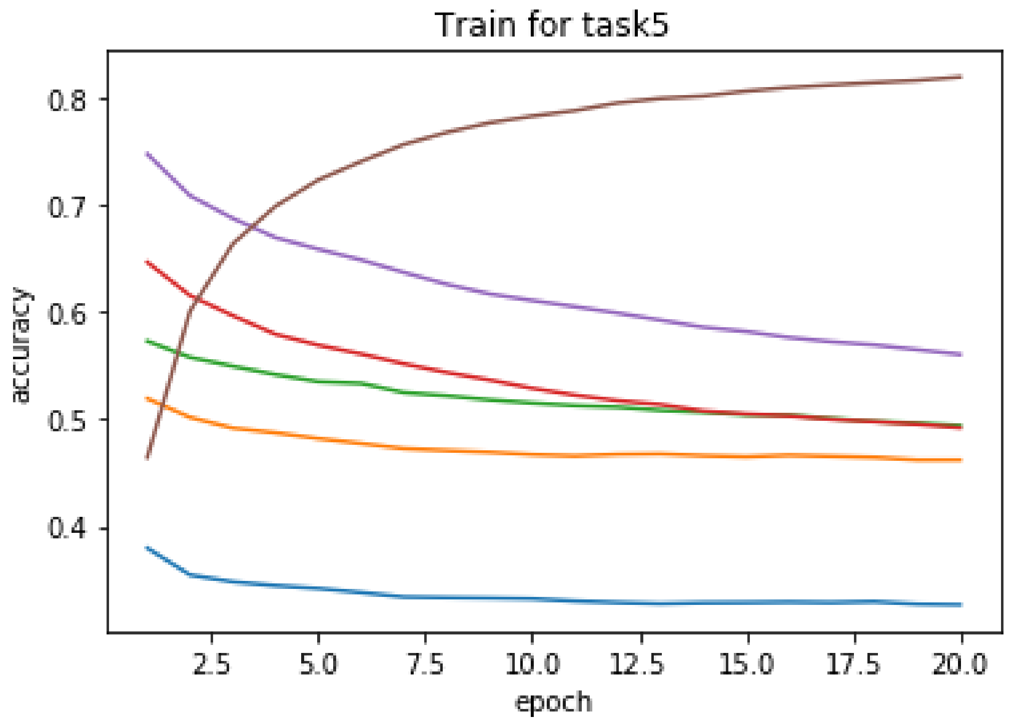

Train for task0
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
accuracy
0
10
20
30
40
50
epoch

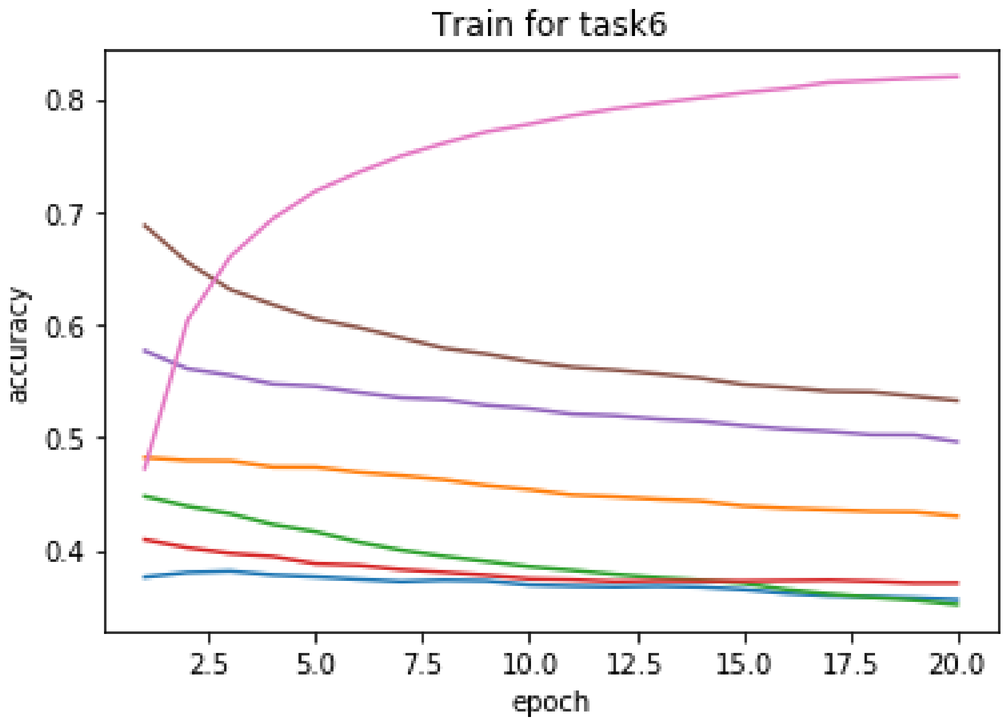

Train for task1
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
accuracy
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
epoch

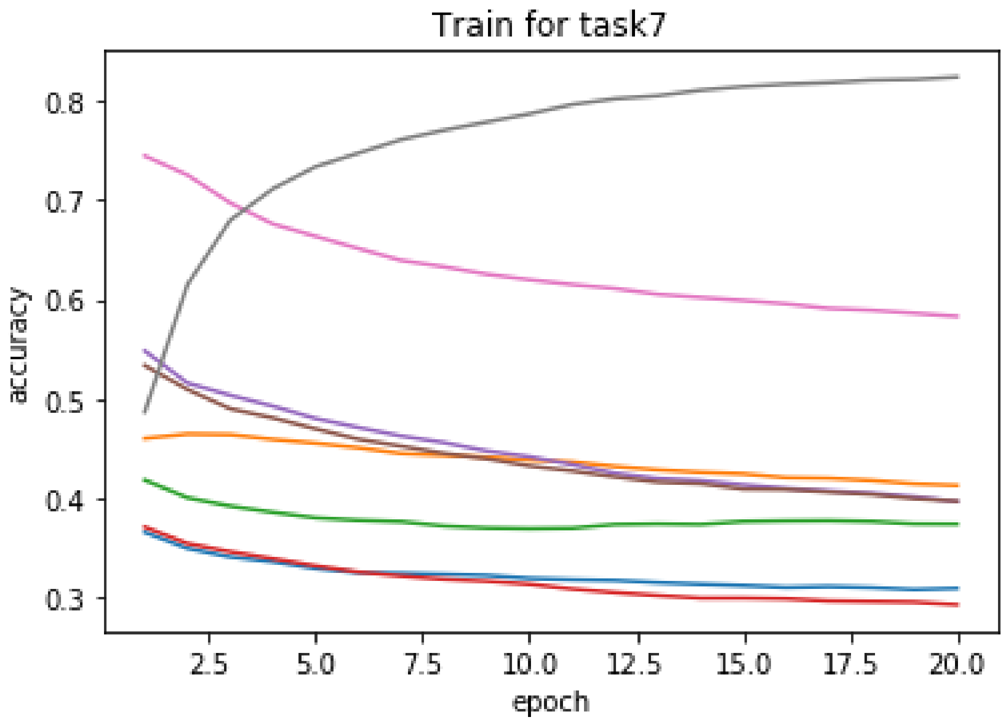

Train for task2
accuracy
epoch
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0

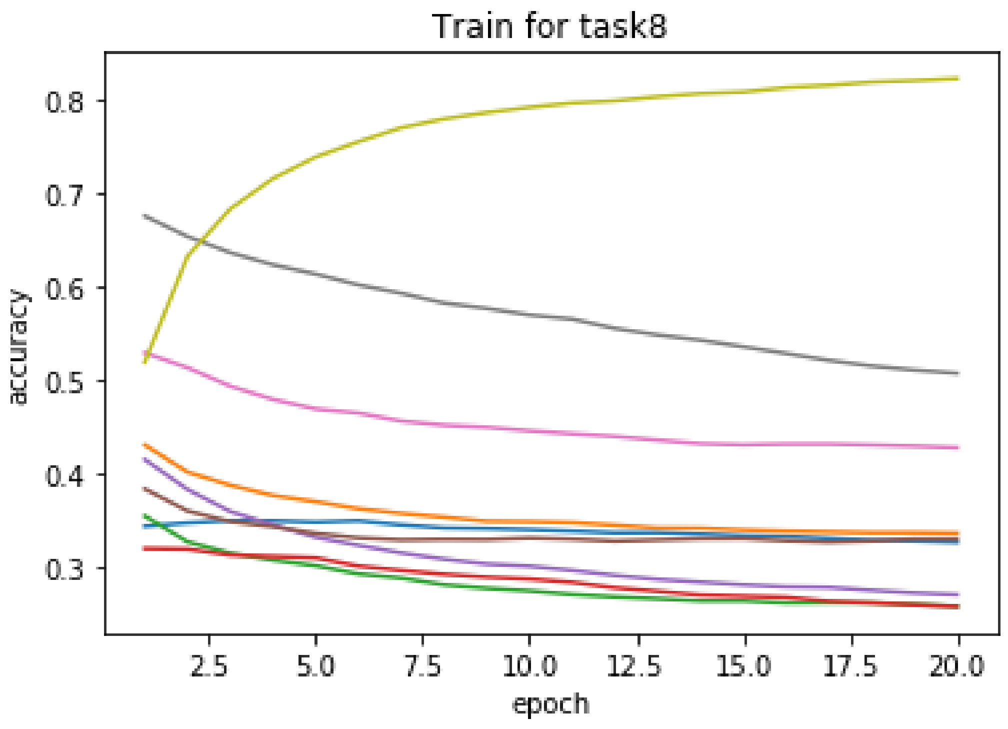

Train for task3
accuracy
epoch
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0.45
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0

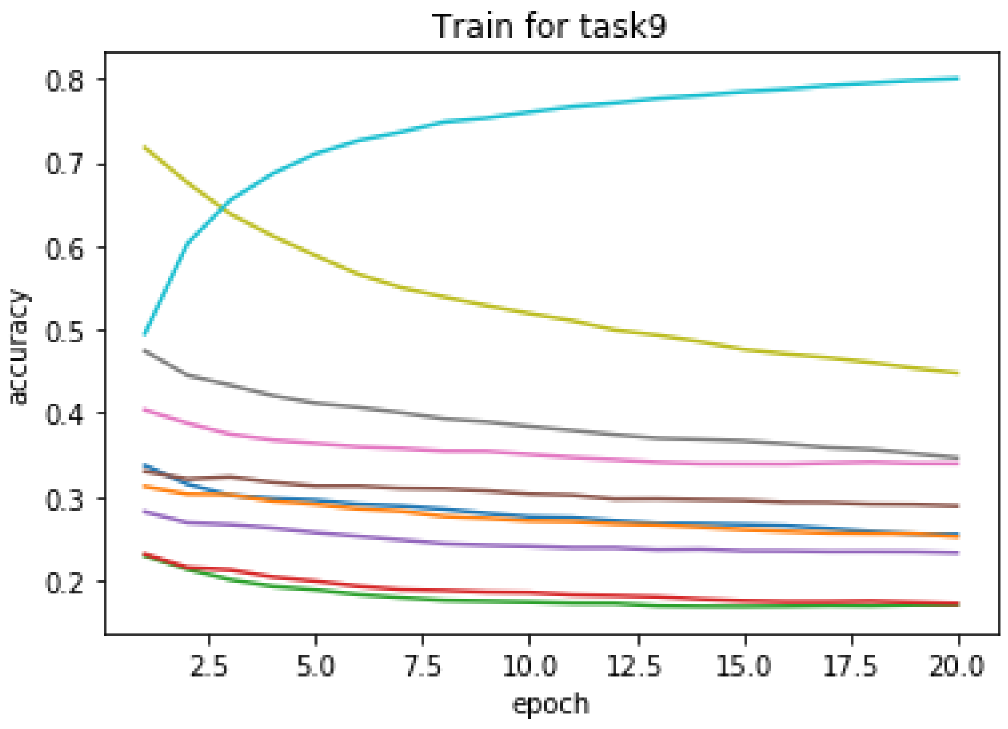

Train for task4
accuracy
epoch
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0.45
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0


Train for task5
accuracy
epoch
0.8
0.7
0.6
0.5
0.4
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0


Train for task6
accuracy
epoch
0.4
0.5
0.6
0.7
0.8
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0


Train for task7
accuracy
epoch
0.3
0.4
0.5
0.6
0.7
0.8
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0

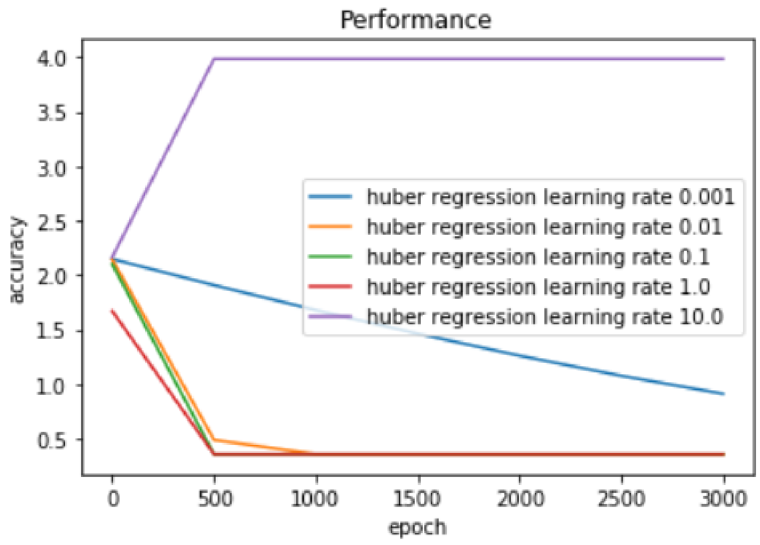




The label for each line was forgot to be put so just follow the plots and you'll interpet what color is for which task. :))

1.10 Changing The learning rate

The learning rate has been changed from 0.001 to 10 each time increasing 10 times. the result is as follows:



1.11 CBWT and TBWT

For 2 layer neural network with gradient decent the result is as follows:

CBWT = [-0.004544450177086724, -0.015700004994869232, -0.029357152325766425, -0.048466662565867104, -0.091020005941391, -0.13772499561309814, -0.21096666653951007, -0.36504998803138733, -0.8019000291824341]

TBWT = -0.5641333442264134

1.12 Effect of GPU

Once again by running the models on gpu we observed that the not only the run time doesn't decrease but also the increase it's so significant that using gpu looks like an old method of running model in comparison to cpu. Still, the reason might be the number of layer we use. Maybe by using more layers we get a better result than we get now.