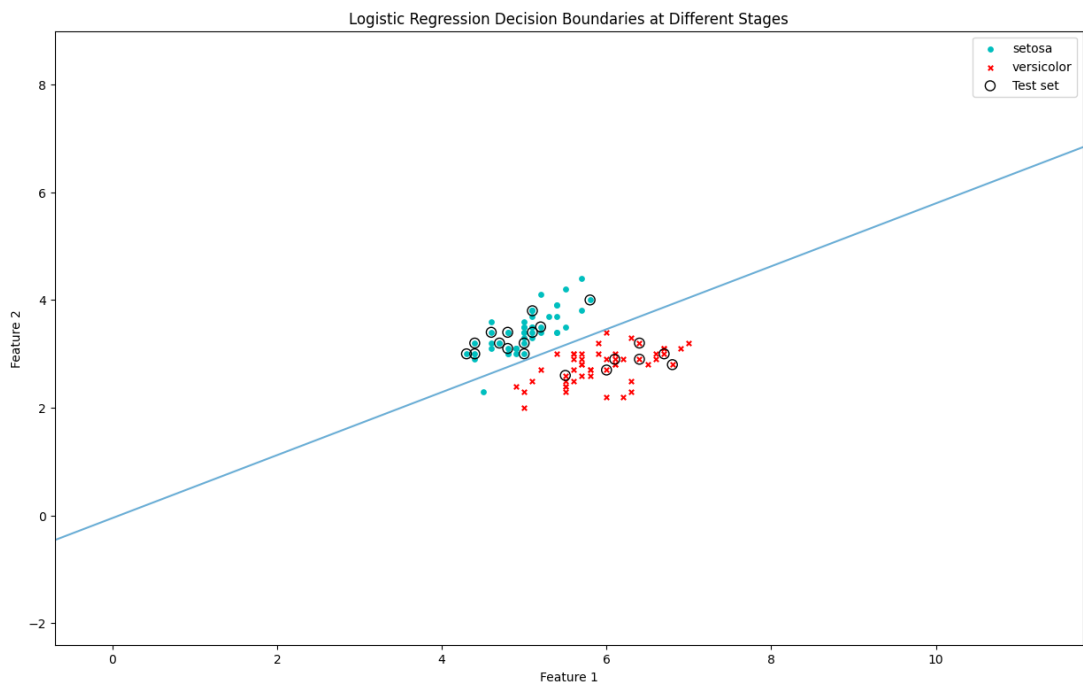


Deep learning lab meeting 1: Final report

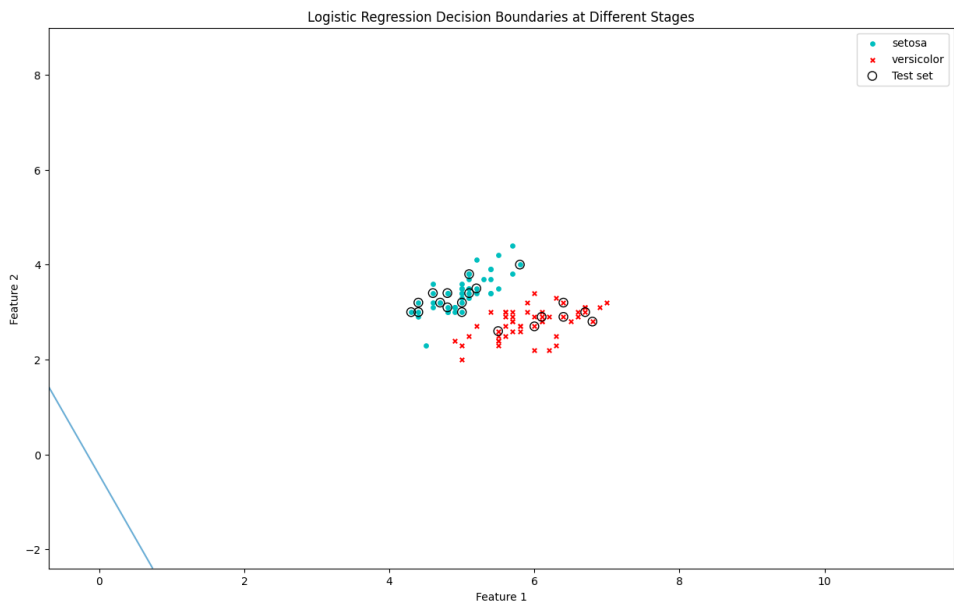
Izar Hasson, Aviv Shem-Tov

lr = 0.5



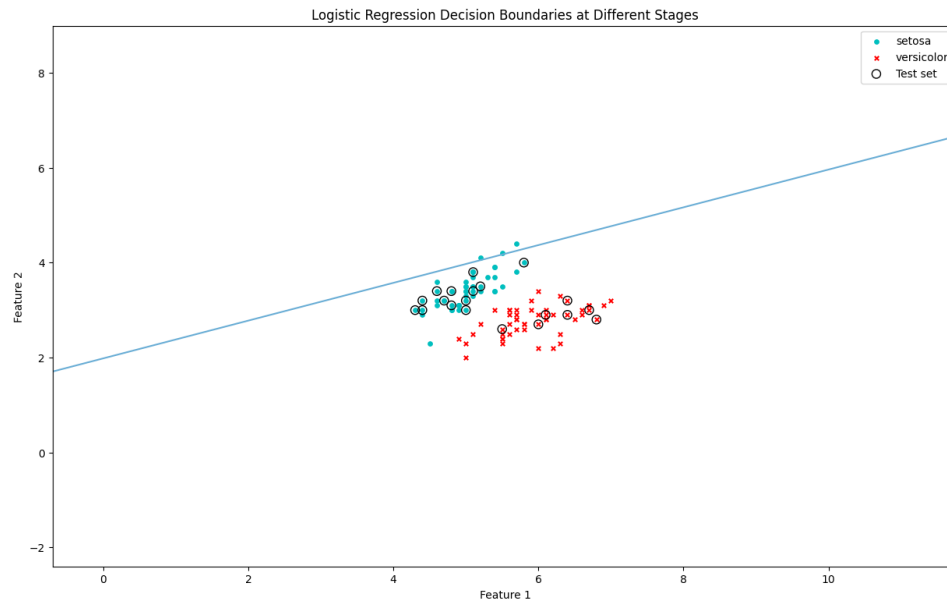
Epoch	Batch	Train Loss	Train Accuracy	Test Loss	Test Accuracy
1	1	1.1	0.46	3.83	0.35
2	1	2.83	0.54	0.77	0.65
3	1	1.12	0.46	3.79	0.35
4	1	2.8	0.54	0.71	0.65
5	1	1.04	0.46	3.6	0.35
6	1	2.66	0.54	0.71	0.65
7	1	1.06	0.46	3.53	0.35
8	1	2.61	0.54	0.66	0.65
9	1	1.0	0.46	3.36	0.35
10	1	2.48	0.54	0.67	0.65
11	1	1.02	0.46	3.26	0.35
12	1	2.41	0.54	0.64	0.65
...					
299	1	0.08	0.99	0.04	1.0
300	1	0.08	0.99	0.04	1.0

Lr= 3



Epoch	Batch	Train Loss	Train Accuracy	Test Loss	Test Accuracy
1	1	0.71	0.54	2.18	0.65
2	1	3.18	0.46	36.5	0.35
3	1	26.98	0.54	36.5	0.35
4	1	26.98	0.54	36.5	0.35
5	1	26.98	0.54	36.5	0.35
6	1	26.98	0.54	36.5	0.35
7	1	26.98	0.54	36.5	0.35
8	1	26.98	0.54	36.5	0.35
9	1	26.98	0.54	36.5	0.35
10	1	26.98	0.54	36.5	0.35
11	1	26.98	0.54	36.5	0.35
12	1	26.98	0.54	36.5	0.35
...					
299	1	26.98	0.54	36.5	0.35
300	1	26.98	0.54	36.5	0.35

Lr=0.05

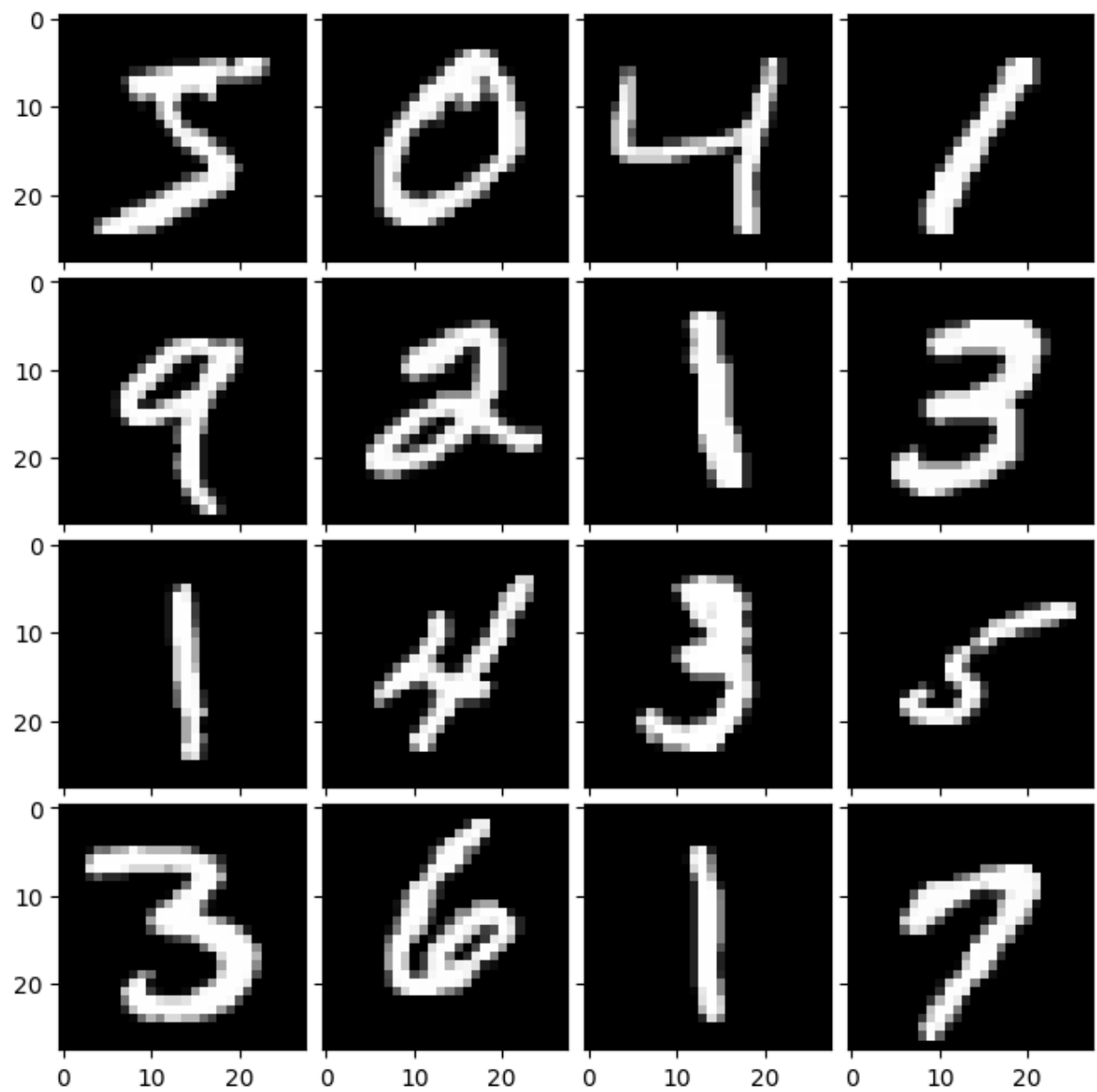


Epoch	Batch	Train Loss	Train Accuracy	Test Loss	Test Accuracy
1	1	1.33	0.46	0.93	0.65
2	1	1.07	0.46	0.92	0.15
3	1	0.96	0.16	0.94	0.0
4	1	0.93	0.01	0.95	0.0
5	1	0.91	0.04	0.96	0.0
6	1	0.91	0.09	0.96	0.0
7	1	0.9	0.12	0.96	0.0
8	1	0.9	0.16	0.95	0.0
9	1	0.89	0.16	0.95	0.0
10	1	0.89	0.16	0.95	0.0
11	1	0.88	0.16	0.94	0.0
12	1	0.88	0.19	0.94	0.0
...					
299	1	0.33	0.99	0.33	1.0
300	1	0.33	0.99	0.32	1.0

1.3) we can see through the table that Lr=0.05 and Lr=0.5 reached a test accuracy of 1 which means they converged correctly, however we can see that Lr= 3 reached accuracy of 0.35 , which means that even after 300 epochs he couldnt converge.

1.4) we can see through the graph that only Lr=0.5 reached a significant improvement, it can separate the data in right way. However the Lr=0.05,3 are nowhere near converging to the right point, but we know that after 250 more the 0.05 managed to, unlike the 3.

2. FCNN classifier



Epoch	Batch	Train Loss	Train Accuracy	Test Loss	Test Accuracy
0	0	2.37	0.11	2.48	0.1
1	938	0.34	0.9	0.19	0.94
2	938	0.16	0.95	0.13	0.96
3	938	0.11	0.97	0.11	0.97
4	938	0.09	0.98	0.09	0.97
5	938	0.07	0.98	0.08	0.97
6	938	0.06	0.98	0.07	0.98
7	938	0.05	0.99	0.07	0.98
8	938	0.04	0.99	0.07	0.98
9	938	0.03	0.99	0.06	0.98
10	938	0.03	0.99	0.06	0.98
11	938	0.03	1.0	0.06	0.98
...					
29	938	0.0	1.0	0.06	0.98
30	938	0.0	1.0	0.06	0.98

We can see that the conversion rate is very fast, even after just one epoch we jump from 0.1 test accuracy (which is equivalent to a complete guess) to 0.94.

That means that our model is really compatible with the data and manages to represent it in a way that is easily separated.



Part 2:

Without augmentations

lr = 0.03

momentum = 0.9 - in the SGDM algorithm, we use momentum to accelerate convergence by adding a fraction of the previous gradient step, thus overcoming local minima and oscillations.

dropout probability = 0.0

epochs = 12

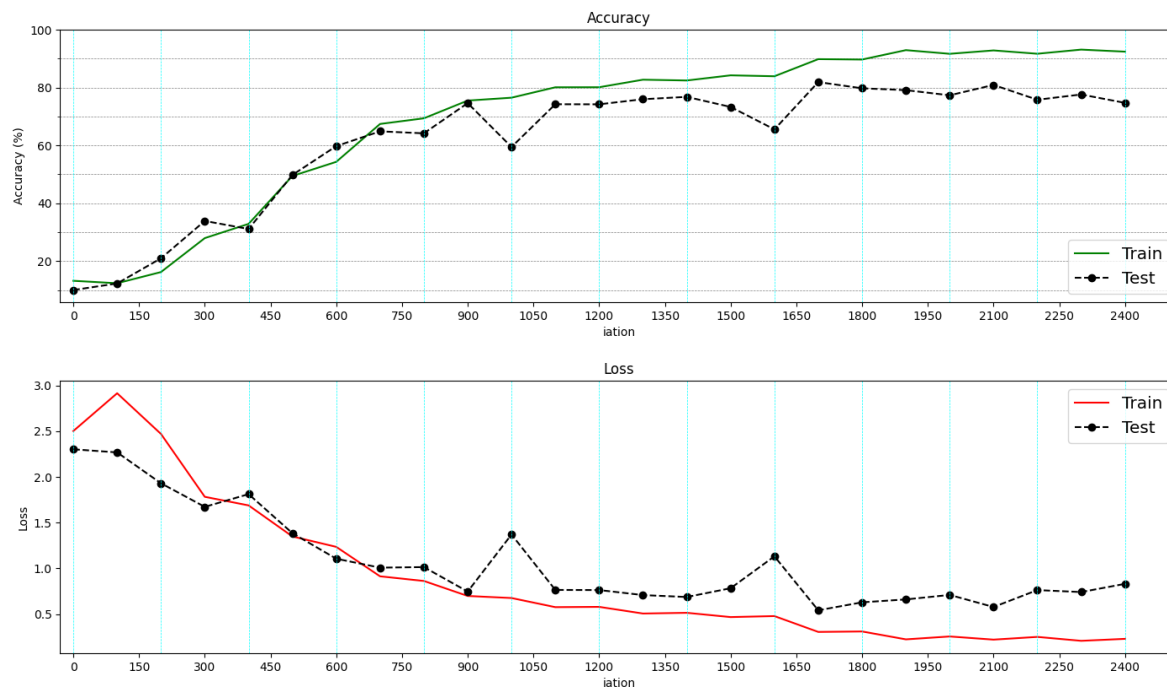
batch size = 250

scheduler step size = 8 - the number of epochs between each scheduler step

scheduler factor = 0.5 - the factor by which the scheduler changes the learning rate

iter break = 100 - number of iterations between validation steps

weight decay = 0.0



75%

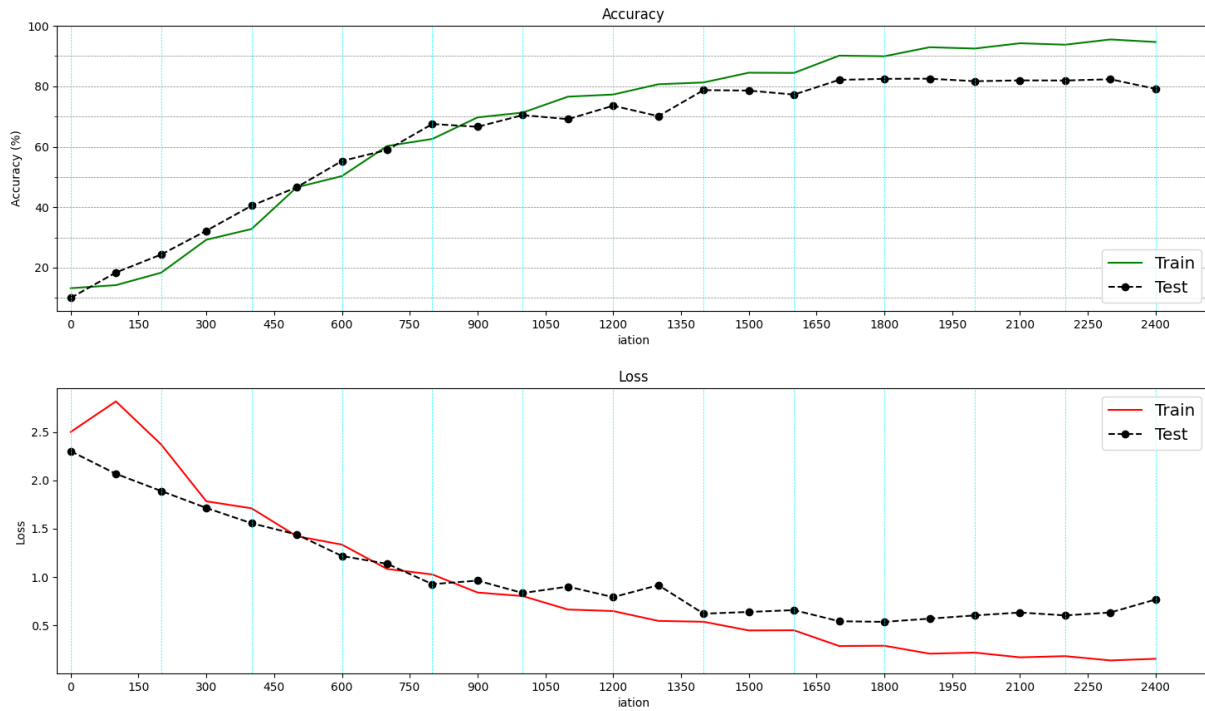
2.2)

We can see the loss trying to converge, but then goes up again, maybe the learning rate at that point is still too big, and the training accuracy is not reaching 1.0.

2.3,4)

We will try to solve the issue by using weight decay:

weight decay = 0.001



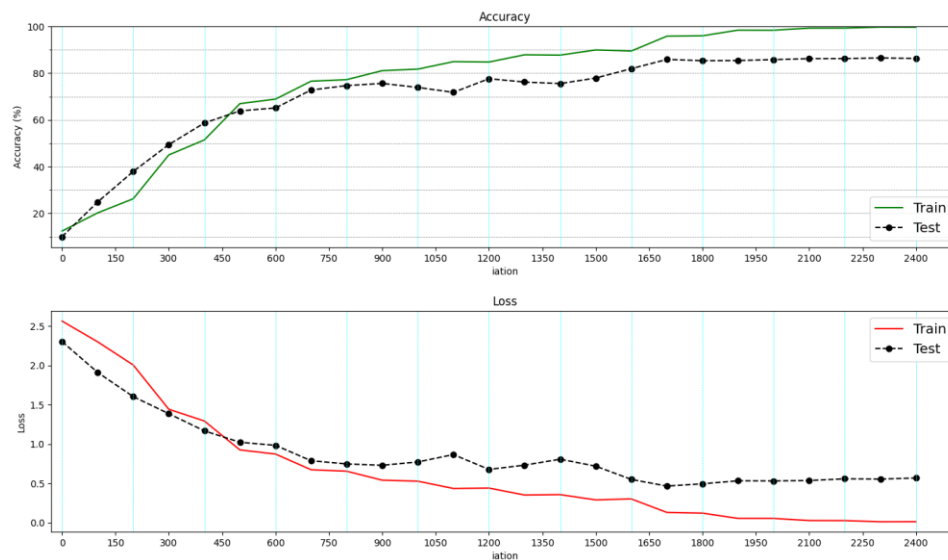
79%

(we skipped a few to be able to finish what's important)

2.5)

scheduler factor = 0.3

dropout probability = 0.15



86%

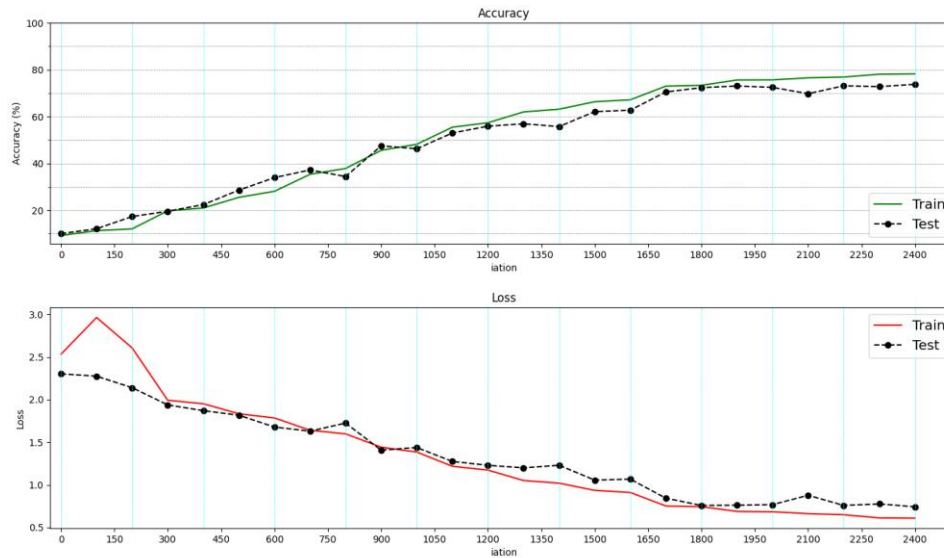
We can see a big increase in test accuracy, notice how the train accuracy reaches 1.0.

2.6)

augmentations = [transforms.RandomHorizontalFlip(p=0.5),

transforms.RandomVerticalFlip(p=0.5)]

Meaning we flip vertically and horizontally with a 50% probability each.



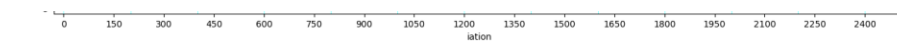
Epoch	Batch	Train Loss	Train Accuracy	Test Loss	Test Accuracy
0	0	2.53	0.09	2.3	0.1
1	100	2.96	0.11	2.28	0.12
1	200	2.61	0.12	2.14	0.17
2	100	1.99	0.2	1.94	0.2
2	200	1.95	0.21	1.87	0.22
3	100	1.83	0.25	1.82	0.29
3	200	1.79	0.28	1.68	0.34
4	100	1.64	0.35	1.63	0.37
4	200	1.6	0.38	1.73	0.34
5	100	1.44	0.46	1.41	0.47
5	200	1.38	0.48	1.44	0.46
6	100	1.22	0.55	1.27	0.53
...
12	100	0.61	0.78	0.78	0.73
12	200	0.61	0.78	0.74	0.74

We can see that the training accuracy was rising but could not reach 1.0, the model needs more training time in order to converge.

We will try to add more augmentations:

```
augmentations = [  
    transforms.RandomVerticalFlip(p=0.5),  
    transforms.RandomRotation(degrees=(-10,10)),  
    transforms.GaussianBlur(kernel_size=(3,3), sigma=(0.1, 1.0)),  
    transforms.ColorJitter(brightness=(0.25,0.75), contrast=(0.25,0.75))]
```

Here we vertically flip each image with probability of 50%, and randomly rotate the image in a degree between -10 and 10 and then blur the image with gaussian blur with a kernel of 3x3 and a sigma of 0.1 and 1.0, then we change the brightness of the image in a random way.



Epoch	Batch	Train Loss	Train Accuracy	Test Loss	Test Accuracy
0	0	2.6	0.1	2.3	0.1
1	100	2.67	0.12	38.97	0.11
1	200	2.45	0.15	55.22	0.1
2	100	2.13	0.2	13.12	0.1
2	200	2.1	0.21	13.38	0.1
3	100	2.02	0.24	2.75	0.18
3	200	1.99	0.25	3.9	0.17
4	100	1.91	0.29	4.26	0.14
4	200	1.9	0.29	2.51	0.26
5	100	1.83	0.32	6.25	0.24
5	200	1.8	0.34	3.03	0.26
6	100	1.7	0.38	2.4	0.29
...					
12	100	1.16	0.58	1.66	0.5
12	200	1.17	0.58	1.79	0.48

We can see that because of the added augmentations the model just couldn't converge within 20 epochs, it simply needs more time.

2.7) so within our limits of 20 epochs, too complicated models just couldn't converge, and we got our best accuracy when we added scheduler factor = 0.3 and dropout

probability = 0.15, we got 86% test accuracy withing 20 epochs, the dropout rate immensely helped the model to converge within a relatively short period of time.

