Introduction to Machine Learning
Assignment 2

by:
Iyad ashraf
7392

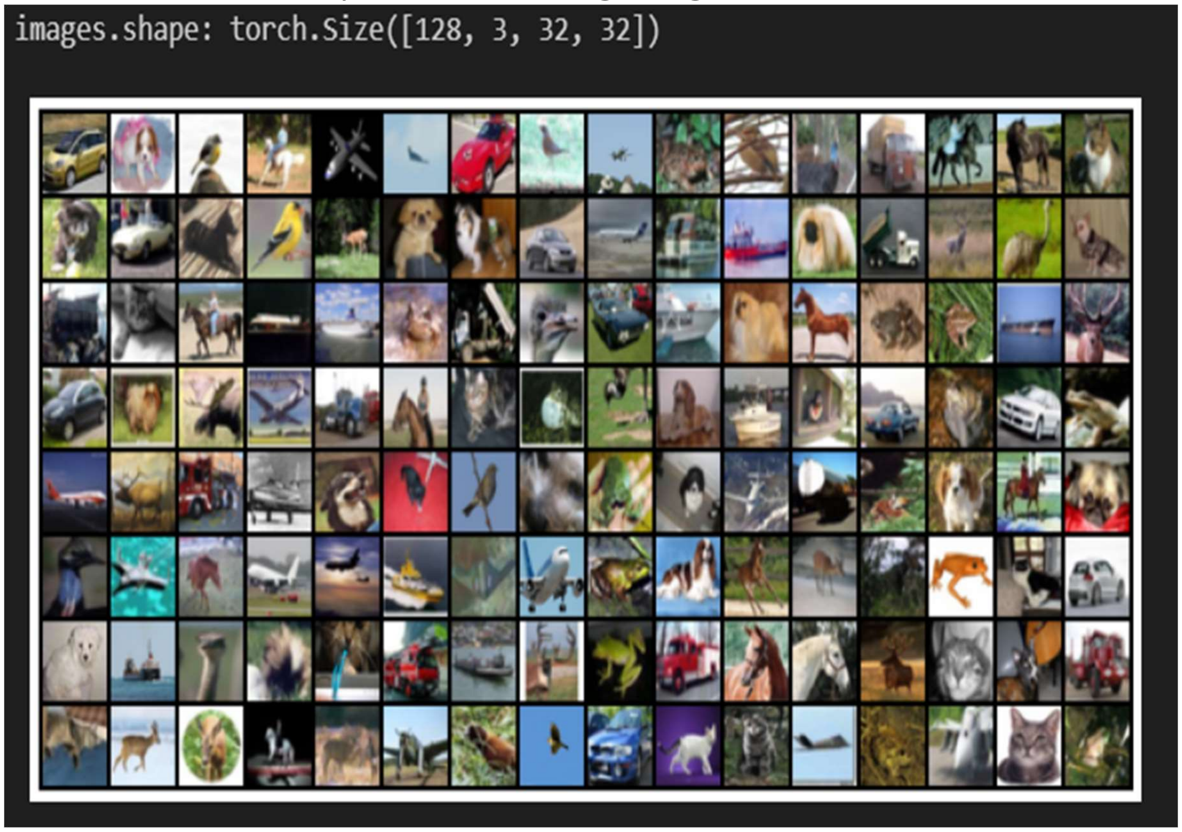
Problem Statement:

Our focus lies on image classification, as we aim to tackle the task of effectively categorizing various images containing different objects such as cars, airplanes, frogs, and more. Our primary objective is to develop a model capable of consistently and accurately identifying the content of each image.

Dataset:

For this task, we will utilize the CIFAR10 dataset, comprising a collection of 60,000 images distributed among ten distinct classes. These classes are numbered from 0 to 9. The dataset is divided into two subsets: the training set consisting of 50,000 images and the test set containing 10,000 images.

Here are some samples of the images figure 1:



```
images.shape: torch.Size([128, 3, 32, 32])
```

Figure 1

Models:

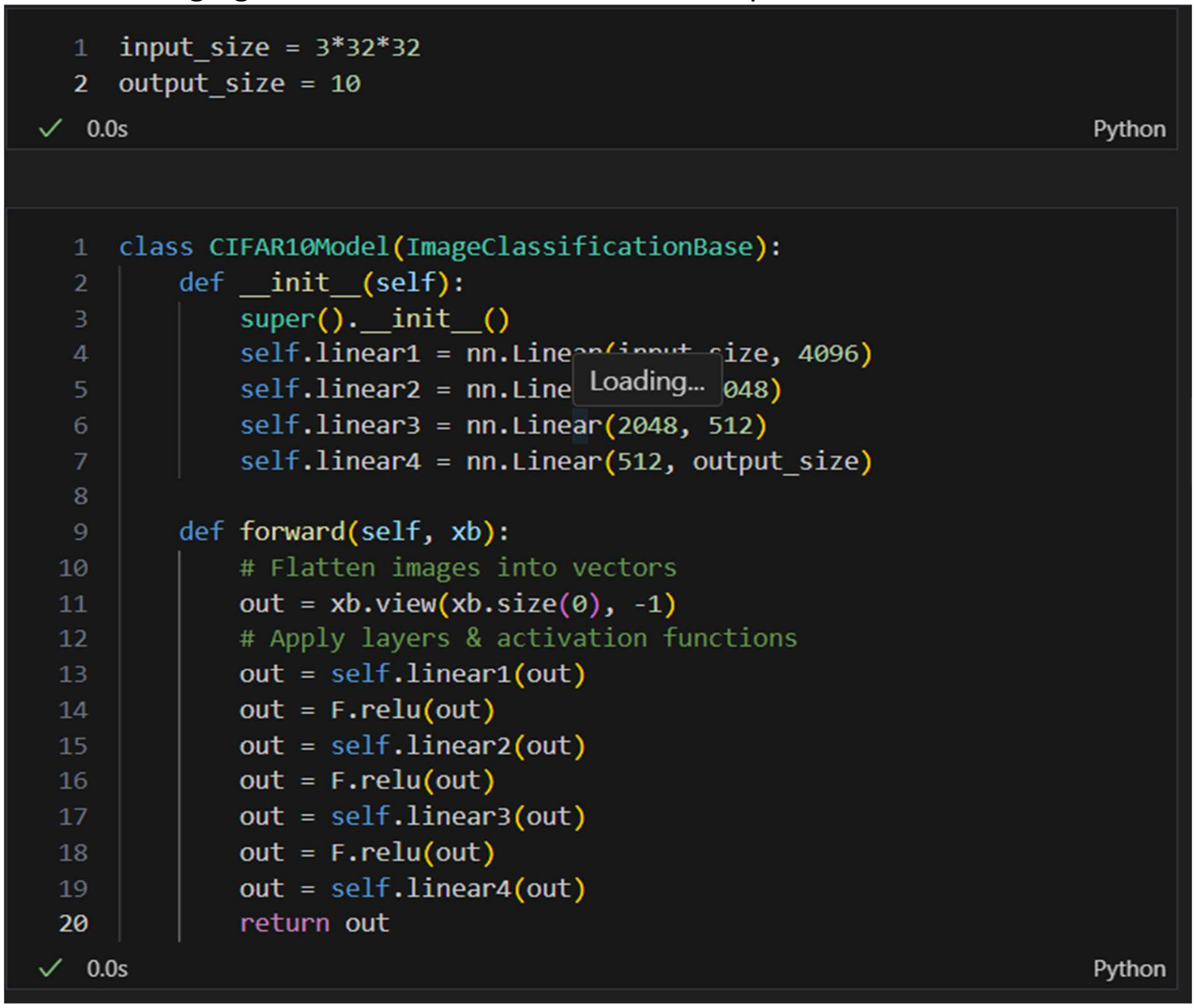
We will train the Dataset using 4 models to analyze and compare the outcomes.

The initial model, which we will create ourselves, will have the following specifications:

- It will consist of 3 Hidden layers.
- The first layer will include 4096 nodes.
- The second layer will have 2084 nodes.
- The third and final layer will contain 512 nodes.
- There will be 10 outputs.
- All layers will utilize the Relu activation function.

Model One:

The following figure 2 will be our model with the specifications mentioned above:



```python
input_size = 3*32*32
output_size = 10
```

✓ 0.0s Python

```python
class CIFAR10Model(ImageClassificationBase):
    def __init__(self):
        super().__init__()
        self.linear1 = nn.Linear(input_size, 4096)
        self.linear2 = nn.Line[illegible]048)
        self.linear3 = nn.Linear(2048, 512)
        self.linear4 = nn.Linear(512, output_size)

    def forward(self, xb):
        # Flatten images into vectors
        out = xb.view(xb.size(0), -1)
        # Apply layers & activation functions
        out = self.linear1(out)
        out = F.relu(out)
        out = self.linear2(out)
        out = F.relu(out)
        out = self.linear3(out)
        out = F.relu(out)
        out = self.linear4(out)
        return out
```

✓ 0.0s Python

Figure 2

Training:

we will train the model on 4 stages of 10 epochs each. This will allow us to save the model after every 10 epochs so in case the model overfits we can revert to an older checkpoint without retraining the model from scratch. Luckily the model did not overfit after 40 epochs, but the test accuracy stopped increasing after the first 25 epochs instead it started to move up and down around 54% accuracy which means the model as on the verge of overfitting, we can see the test accuracy reached 54.4% after training which means the model is not generalizing very well but 55% was still the best accuracy achieved through the training. We can conclude from this that the model is not complex enough to catch the patterns in the data and therefore this is the best it can do right now.

Here is the loss plotted against every epoch trained figure 3:

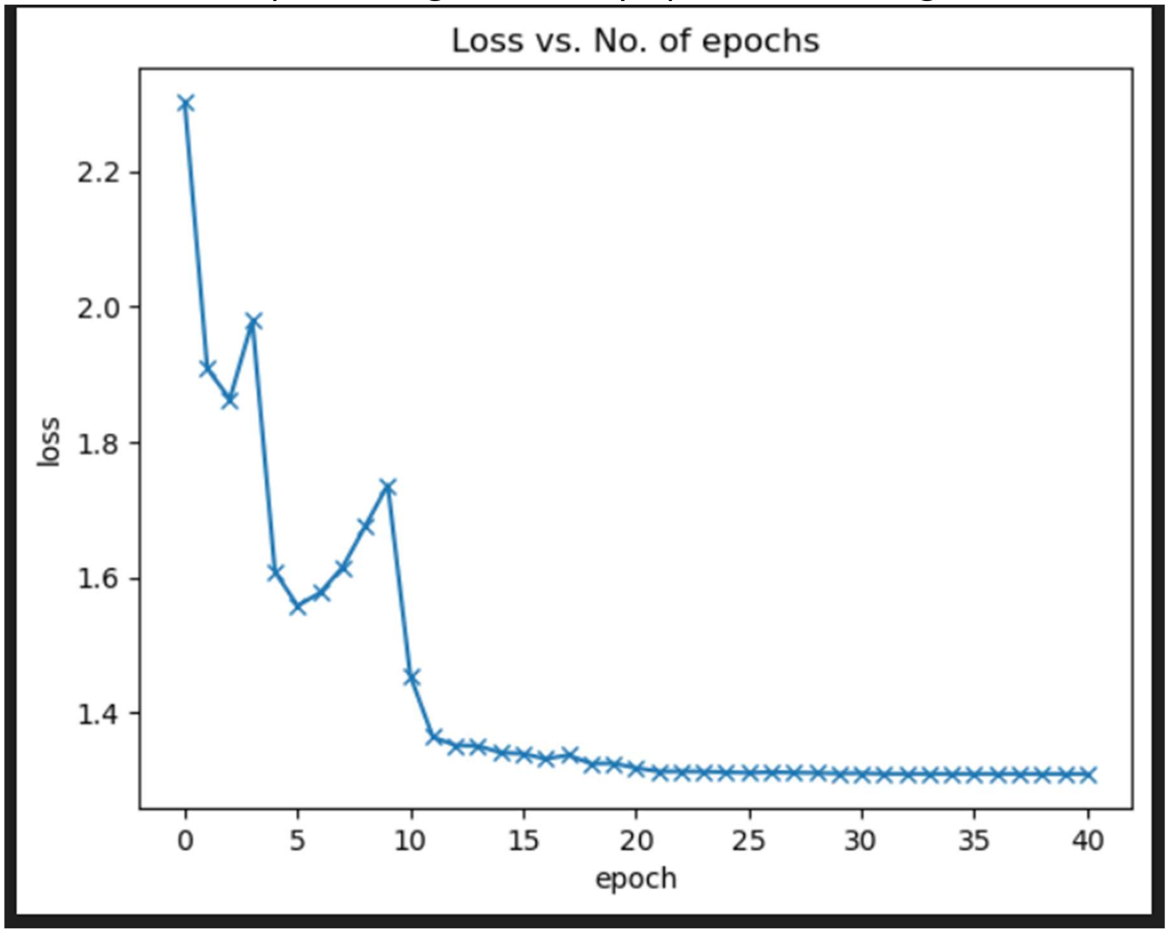


Figure 3

the loss is calculated per batch that's why it is not smoothly decreasing but instead its fluctuating but overall, it does decrease well with training.

Here is the test accuracy plotted at the same rate figure 4:

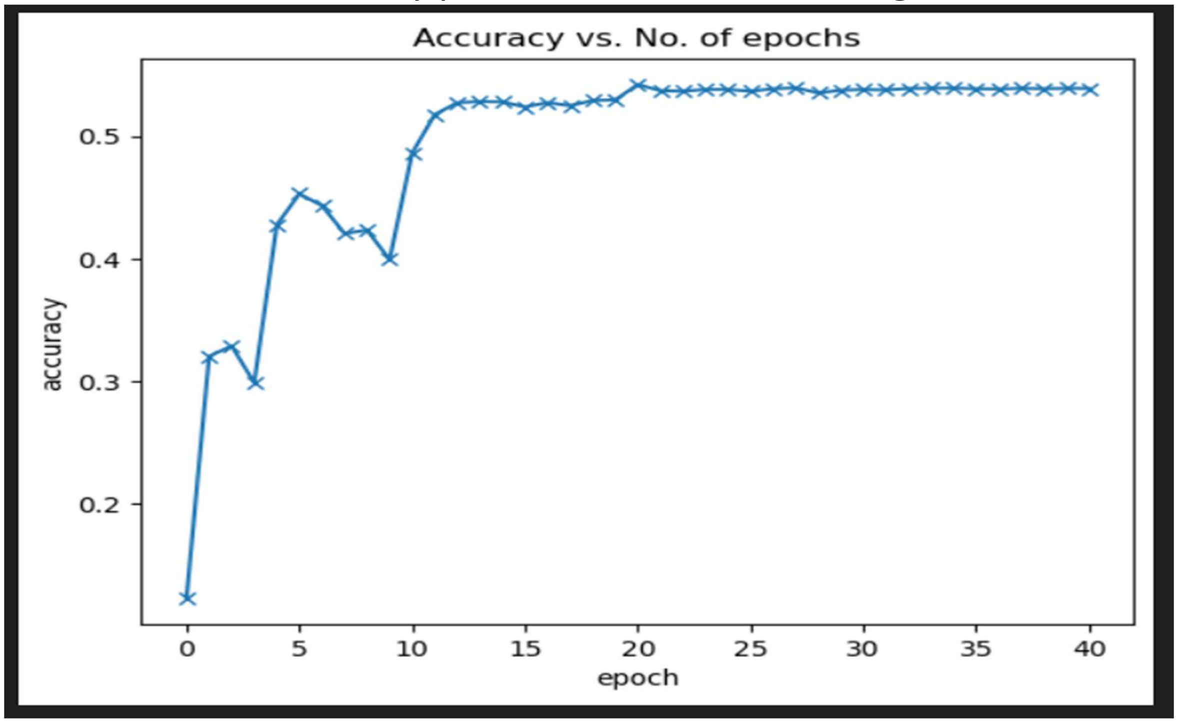
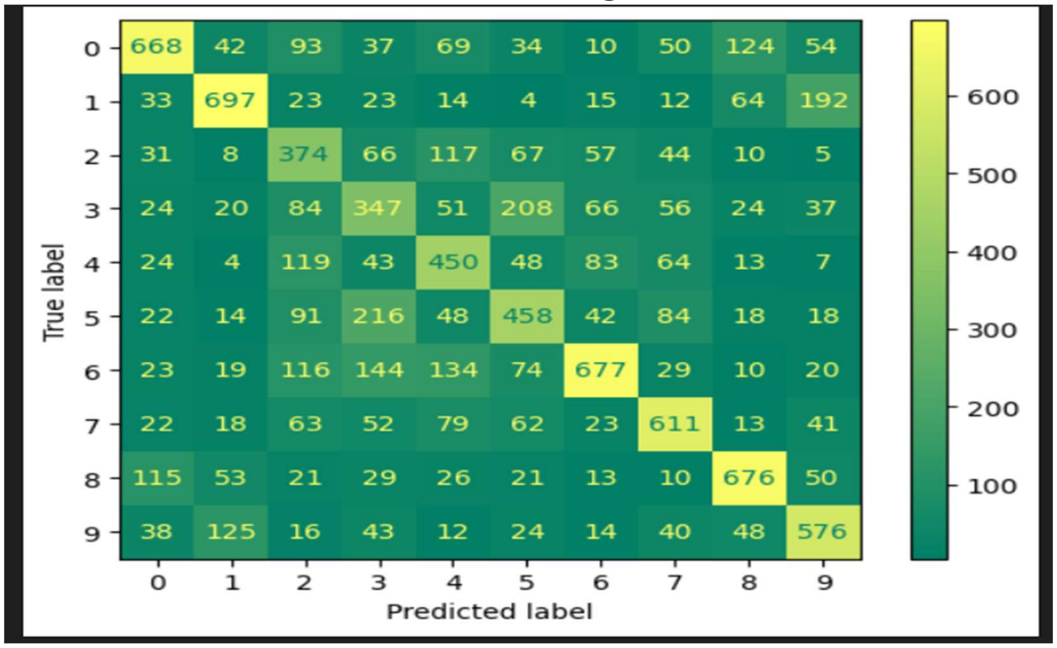


Figure 4

we see that the test accuracy does increases until it reaches about 54% then it starts fluctuating around the same point without really changing much even though the loss was decreasing more and more.
And here is the confusion matrix figure 5:



True label \ Predicted label	0	1	2	3	4	5	6	7	8	9
0	668	42	93	37	69	34	10	50	124	54
1	33	697	23	23	14	4	15	12	64	192
2	31	8	374	66	117	67	57	44	10	5
3	24	20	84	347	51	208	66	56	24	37
4	24	4	119	43	450	48	83	64	13	7
5	22	14	91	216	48	458	42	84	18	18
6	23	19	116	144	134	74	677	29	10	20
7	22	18	63	52	79	62	23	611	13	41
8	115	53	21	29	26	21	13	10	676	50
9	38	125	16	43	12	24	14	40	48	576

Figure 5

It is interesting to observe that among the three classes, 1, 6, and 8, the model demonstrated the highest accuracy in identifying them correctly. On the other hand, classes 2, 3, and 4 proved to be the most challenging for the model to classify accurately. This finding adds an intriguing aspect to the analysis.
Here is a visualization of the image before and after training figure 6:

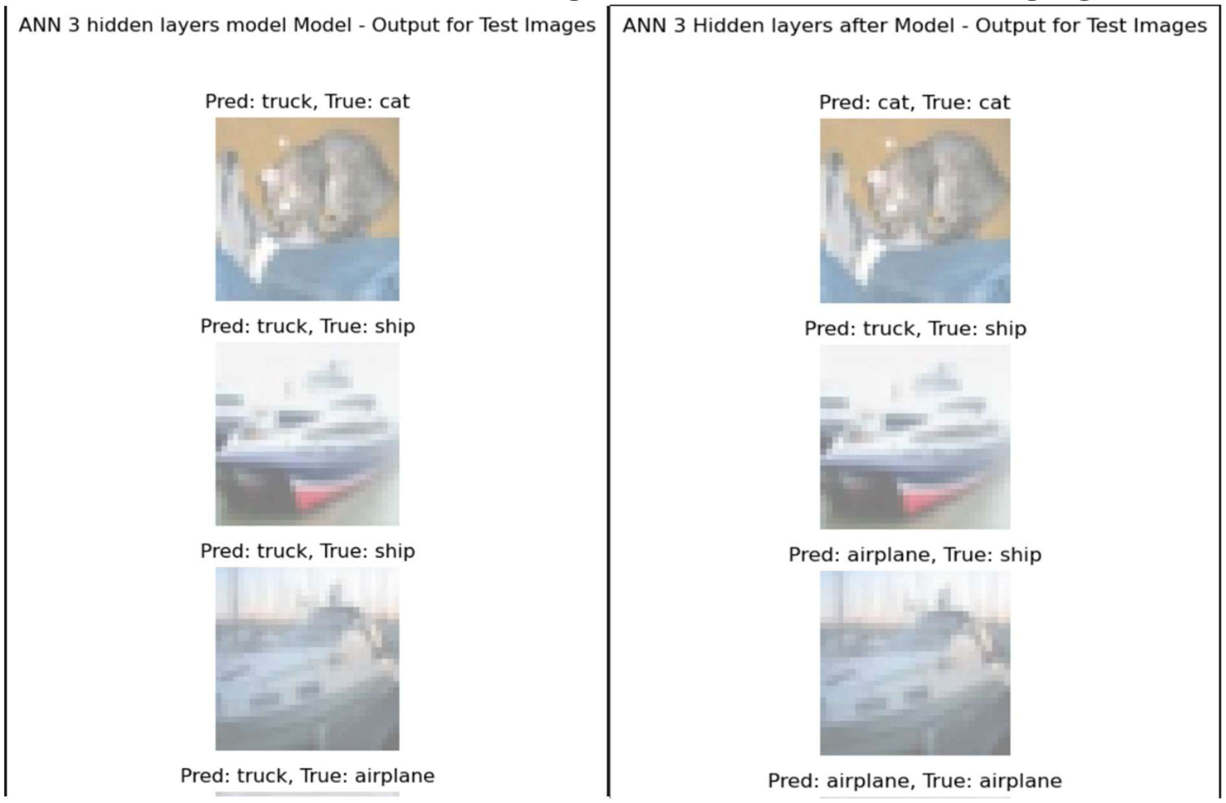

ANN 3 hidden layers model Model - Output for Test Images
Pred: truck, True: cat
Pred: truck, True: ship
Pred: truck, True: ship
Pred: truck, True: airplane
ANN 3 Hidden layers after Model - Output for Test Images
Pred: cat, True: cat
Pred: truck, True: ship
Pred: airplane, True: ship
Pred: airplane, True: airplane

Figure 6

VGG16 Model:

The utilization of pre-trained weights in the model resulted in a remarkable increase in accuracy, elevating it from 10% to 79% after just a single epoch. This exponential improvement surpassed the progress achieved by our model trained from scratch. Notably, after only 10 epochs, the final test accuracy reached an impressive 79%. It is remarkable that the model demonstrated strong generalization capabilities and avoided overfitting, even after attaining such high accuracy on the training data.

Here is the loss plotted against every epoch trained figure 7:

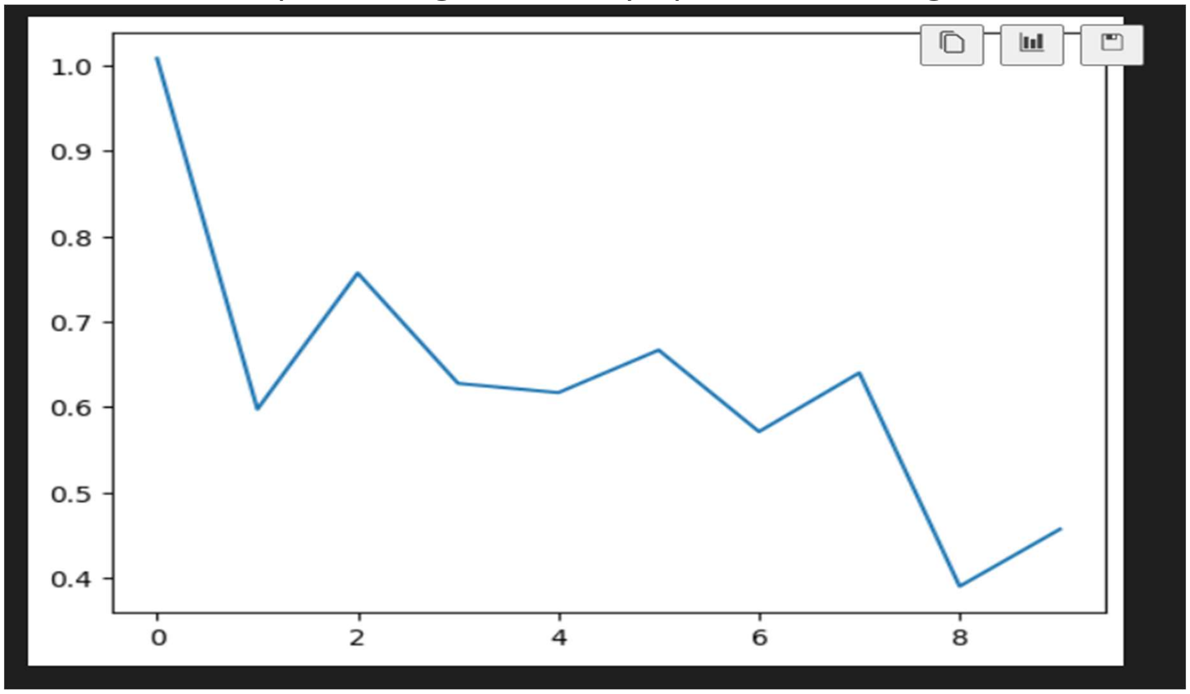


Figure 7

And here is the accuracy plotted figure 8:

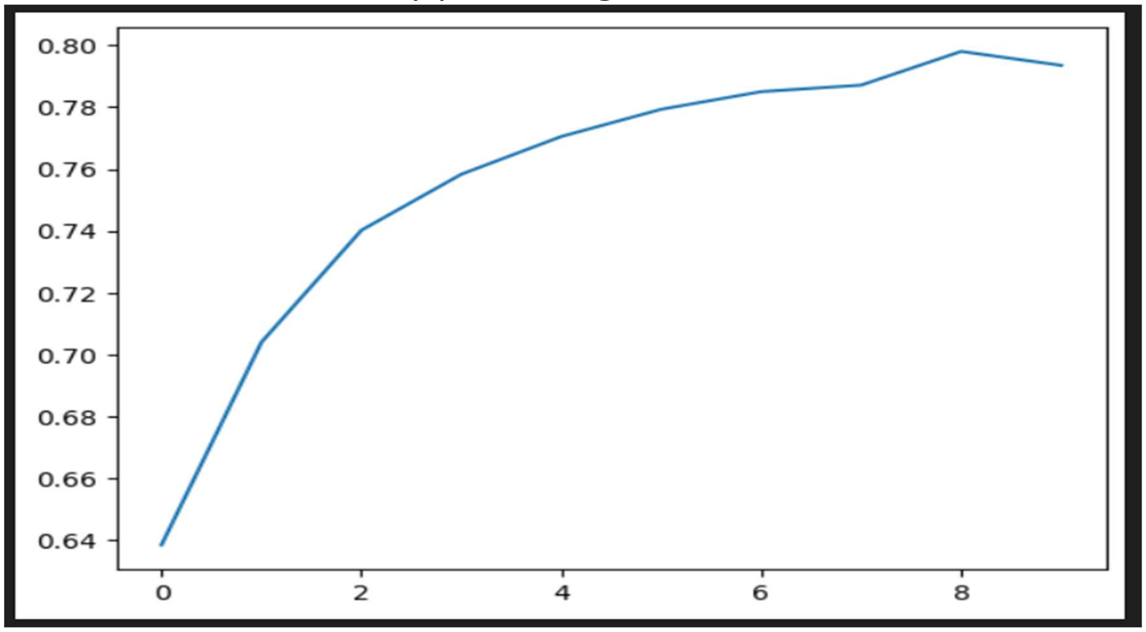
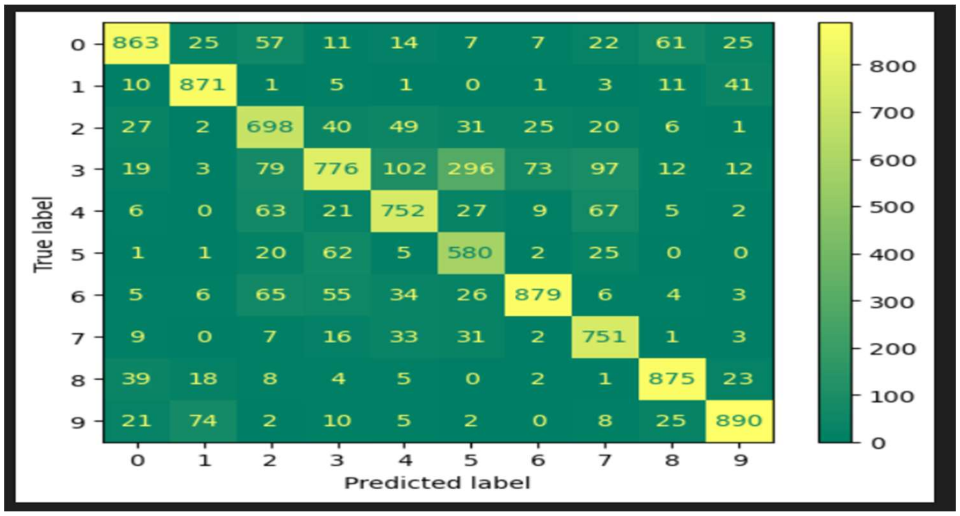


Figure 8

we can see that we could have trained for only 6 epoch and get almost the same results.
Here is the confusion matrix figure 9:



	0	1	2	3	4	5	6	7	8	9
0	863	25	57	11	14	7	7	22	61	25
1	10	871	1	5	1	0	1	3	11	41
2	27	2	698	40	49	31	25	20	6	1
3	19	3	79	776	102	296	73	97	12	12
4	6	0	63	21	752	27	9	67	5	2
5	1	1	20	62	5	580	2	25	0	0
6	5	6	65	55	34	26	879	6	4	3
7	9	0	7	16	33	31	2	751	1	3
8	39	18	8	4	5	0	2	1	875	23
9	21	74	2	10	5	2	0	8	25	890

Figure 9

It is interesting to note that both our model and the alternative model excelled in classifying the same three classes and another class was also classified with good accuracy this shows that by increasing the complexity of model the accuracy increase. These classes proved to be relatively easier for both models to identify correctly. Similarly, both models encountered challenges when it came to classifying the same three classes. This observation adds another intriguing aspect to the comparison between the two models.

Here is a visualization of the image before and after training figure 10:

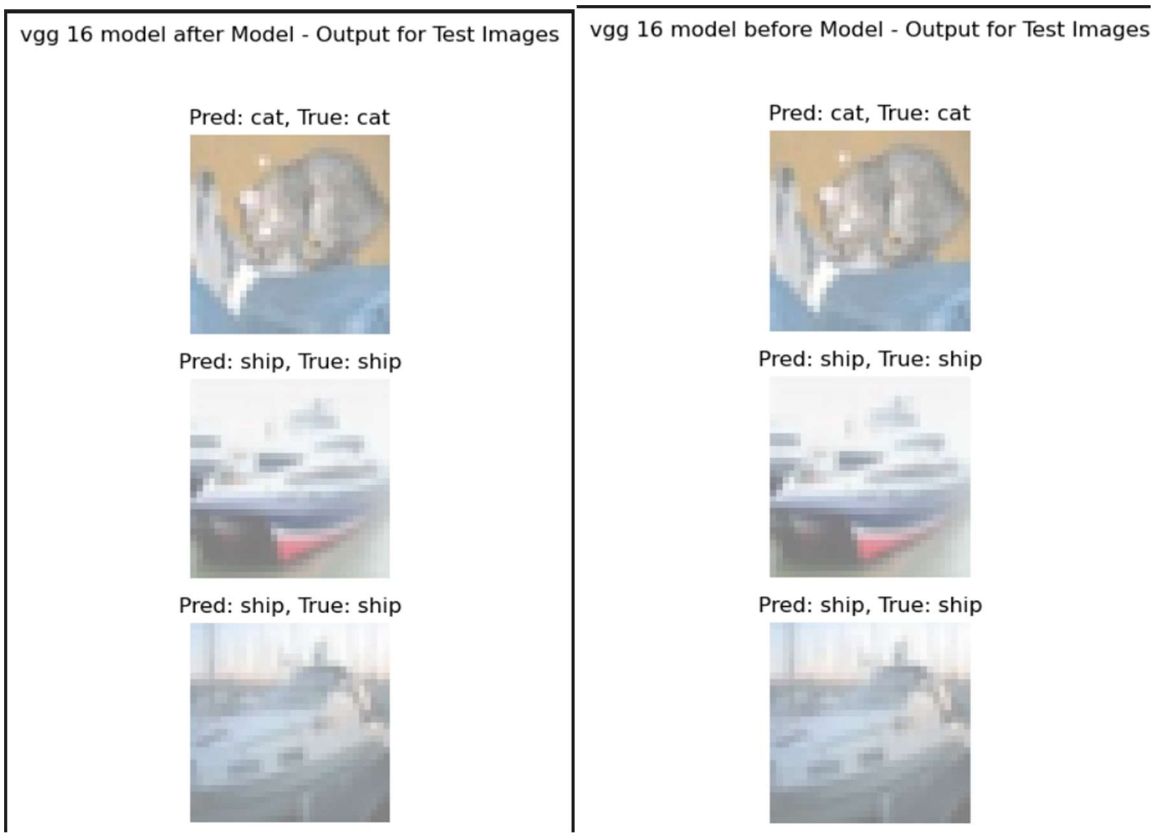


Figure 10

RESNET50 model:

we can see that just like vgg16 this model got great results of 78% test accuracy after just 6 epochs thanks to its pre-trained weights.

Here are the accuracies after 10 epochs of training figure 11:

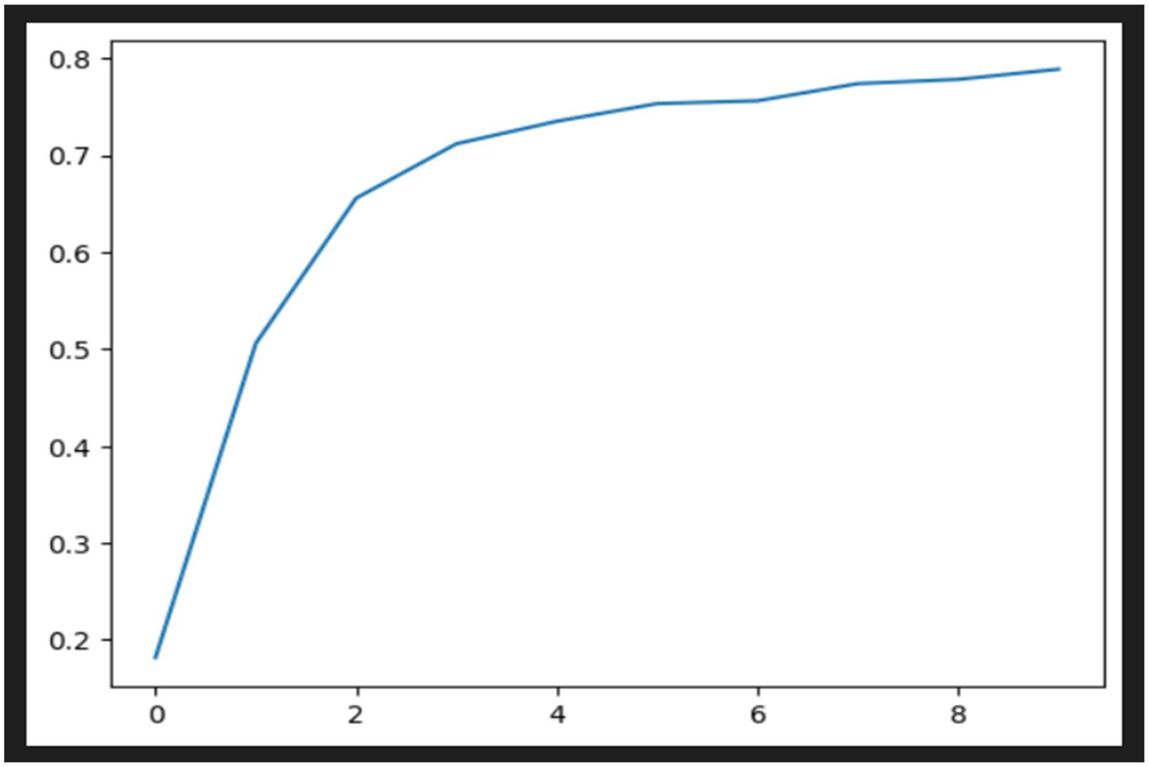


Figure 11

we can see this model did a little less than vgg16 where it got 78% accuracy on the test data.
Here is the loss plotted against every epoch trained figure 12:

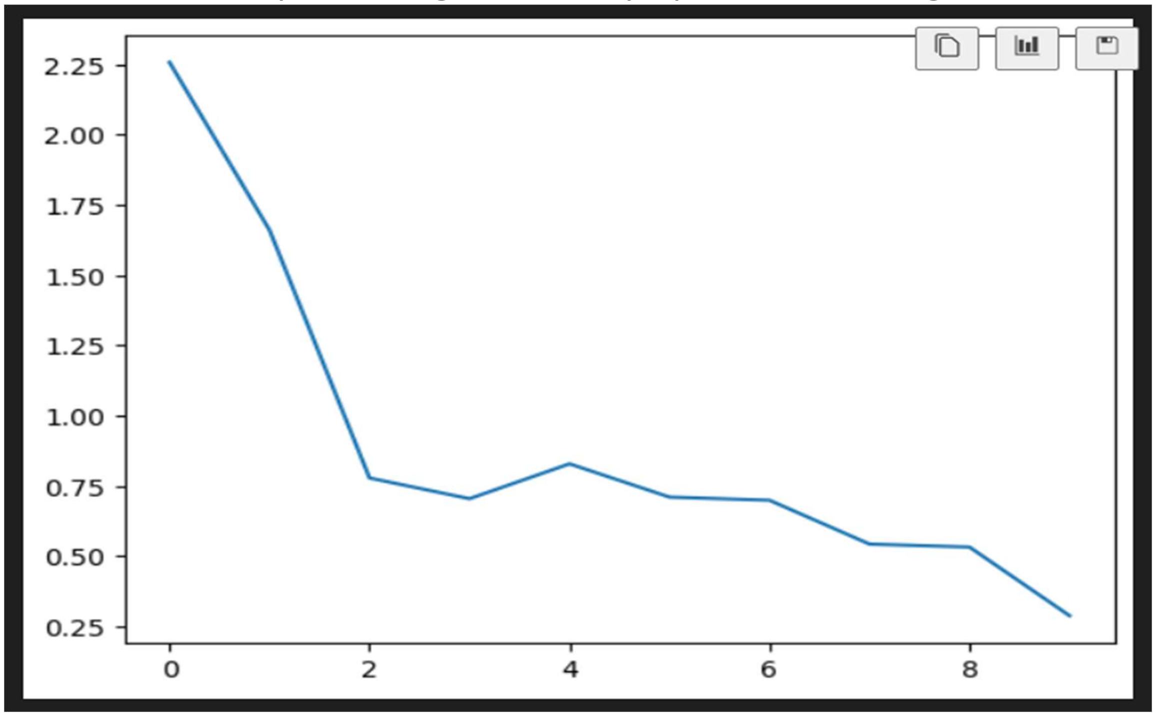
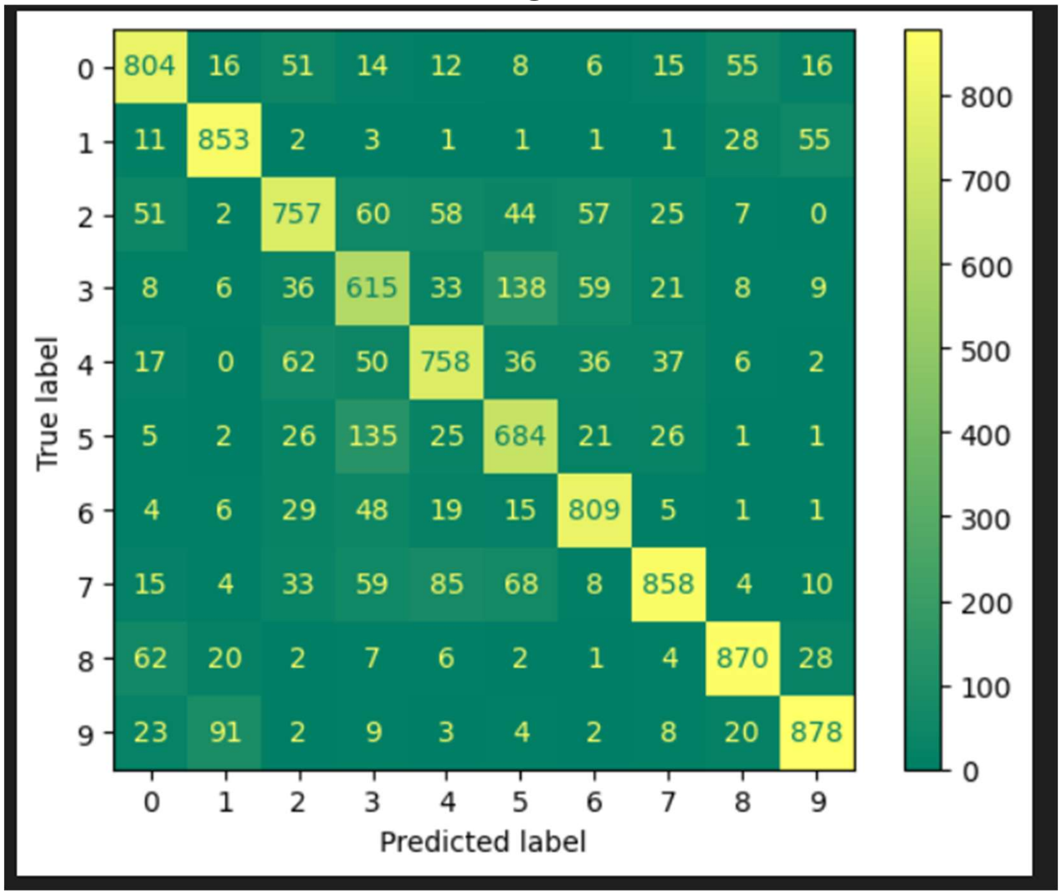


Figure 12

we can see the model was training well and the loss was decreasing.

we can see that this model trained even faster than the vgg16. We noticed that the loss was more smothering than vgg16 lose this could be due to the random data used or the learning rate used in the stochastic gradient descent, but both reach the same accuracy and loss levels.
Here is the confusion matrix figure 13:



True label \ Predicted label	0	1	2	3	4	5	6	7	8	9
0	804	16	51	14	12	8	6	15	55	16
1	11	853	2	3	1	1	1	1	28	55
2	51	2	757	60	58	44	57	25	7	0
3	8	6	36	615	33	138	59	21	8	9
4	17	0	62	50	758	36	36	37	6	2
5	5	2	26	135	25	684	21	26	1	1
6	4	6	29	48	19	15	809	5	1	1
7	15	4	33	59	85	68	8	858	4	10
8	62	20	2	7	6	2	1	4	870	28
9	23	91	2	9	3	4	2	8	20	878

Figure 13

we also see the same pattern here of classifying 1, 6 and 8 better than 2, 3 and 5 and the a new class was introduced being better than them as in vgg16 which is class 9.
Here is a visualization of the image before and after training figure 14:

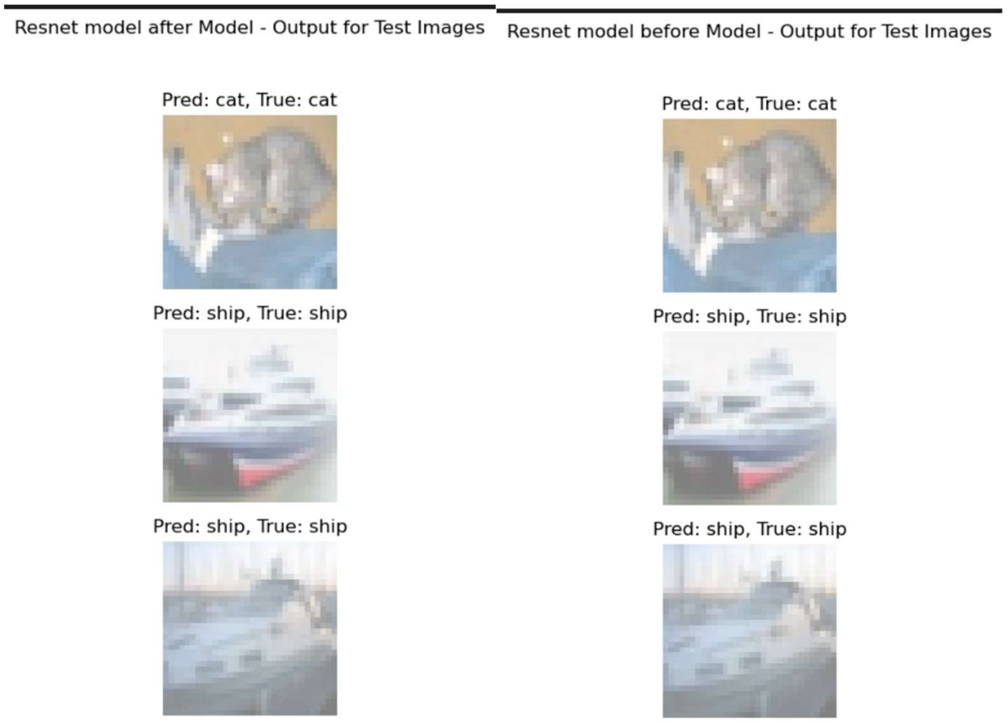


Figure 14

Googlenet:
we can see that this model like the 2 before it also was training pretty fast.
Here is the final accuracy after 10 epochs figure 15:

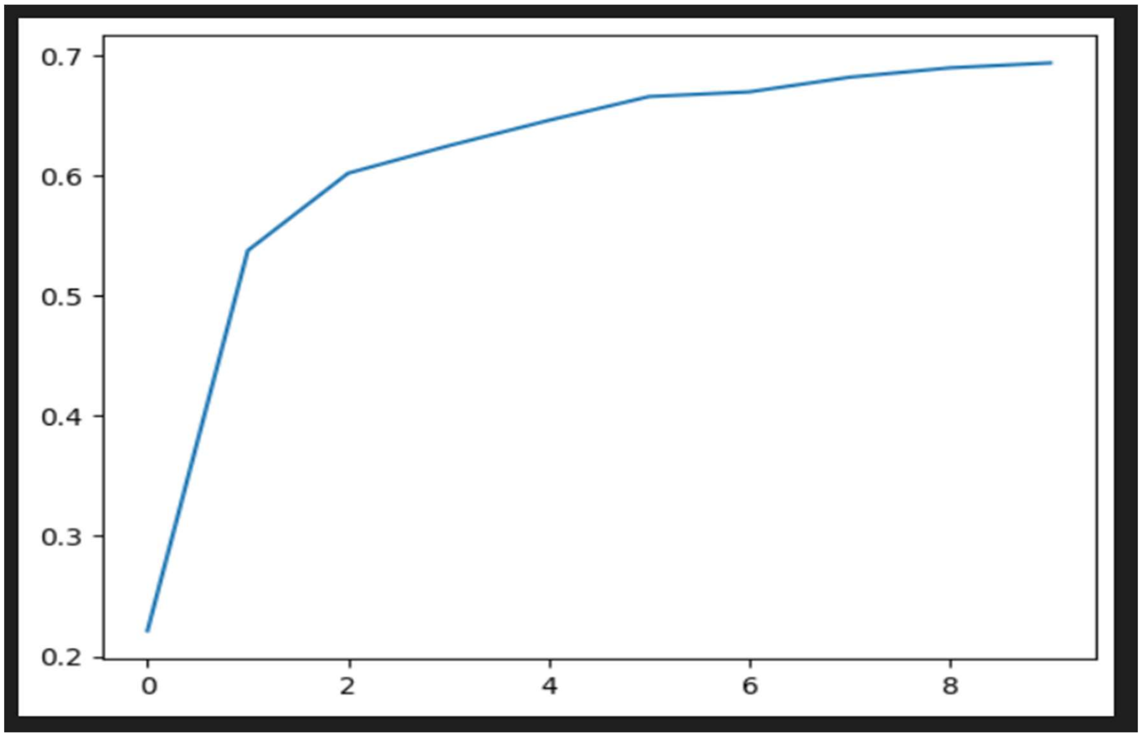


Figure 15

the model did as well as resnet50 or vgg16 as it reached 78% accuracy. Here is the loss graph figure 16:

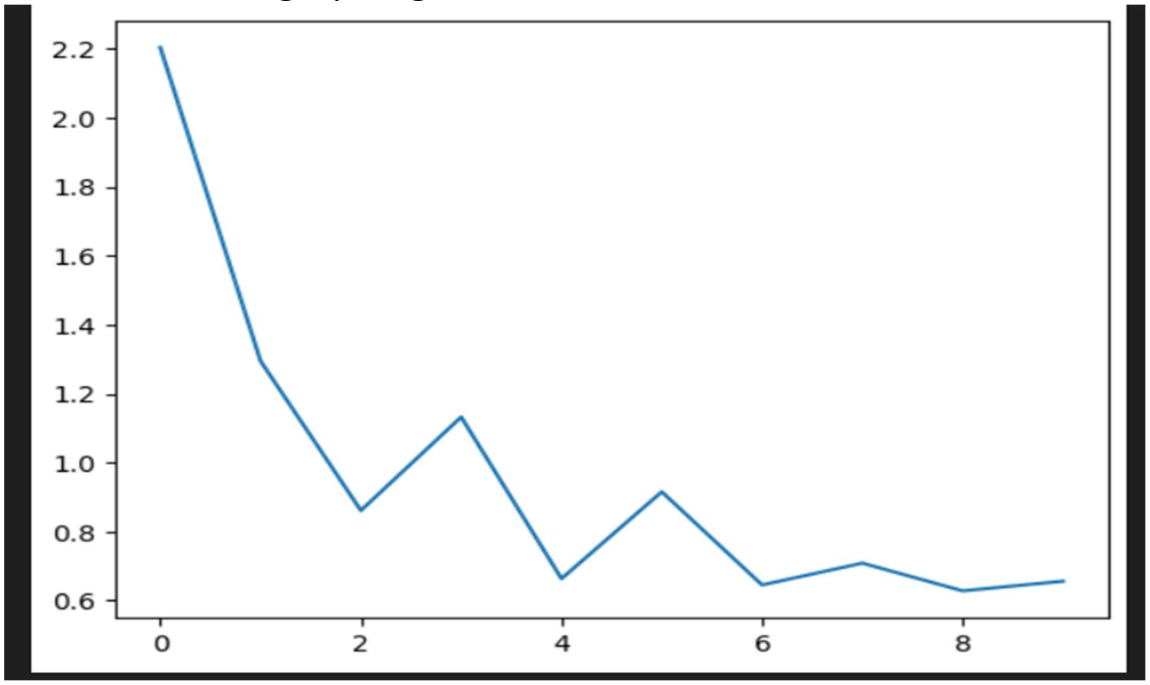
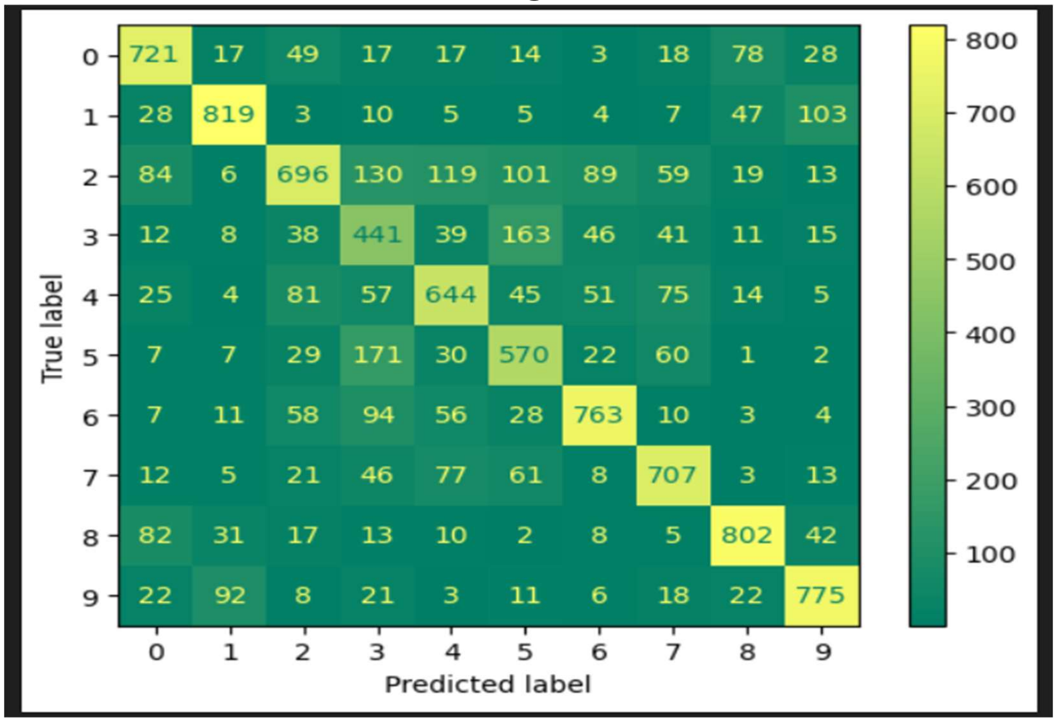


Figure 16

the model was training well, and the loss was decreasing. we can see this model trained did as well as the previous 2 models.
Here is the confusion matrix figure 17:



True \ Predicted	0	1	2	3	4	5	6	7	8	9
0	721	17	49	17	17	14	3	18	78	28
1	28	819	3	10	5	5	4	7	47	103
2	84	6	696	130	119	101	89	59	19	13
3	12	8	38	441	39	163	46	41	11	15
4	25	4	81	57	644	45	51	75	14	5
5	7	7	29	171	30	570	22	60	1	2
6	7	11	58	94	56	28	763	10	3	4
7	12	5	21	46	77	61	8	707	3	13
8	82	31	17	13	10	2	8	5	802	42
9	22	92	8	21	3	11	6	18	22	775

Figure 17

We can still see the same pattern we saw in all models.
Here is a visualization of the image before and after training figure 18:

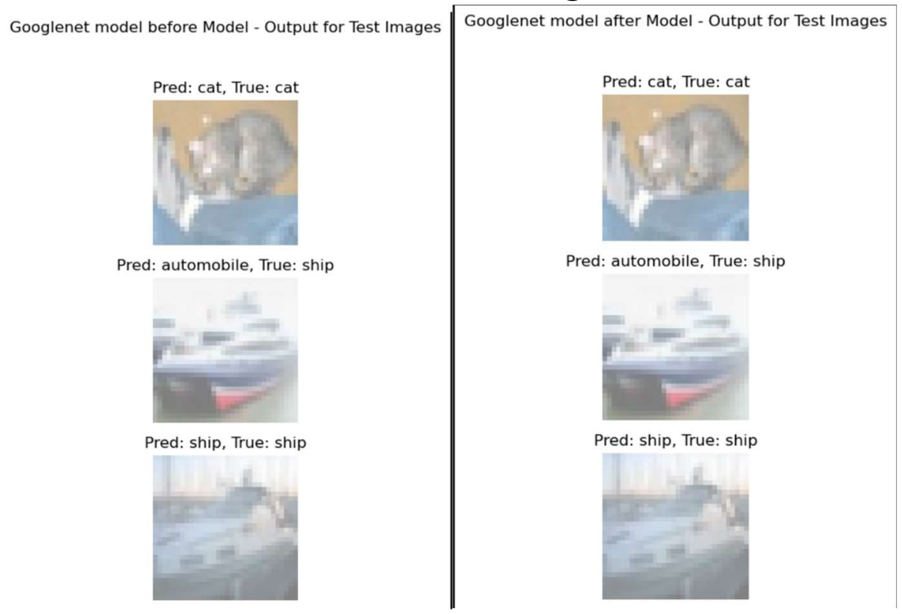

Googlenet model before Model - Output for Test Images
Pred: cat, True: cat
Pred: automobile, True: ship
Pred: ship, True: ship
Googlenet model after Model - Output for Test Images
Pred: cat, True: cat
Pred: automobile, True: ship
Pred: ship, True: ship

Figure 18

Conclusion

The vgg16 model preformed best where it was the most accurate model followed by restnet googleNet and lastly our model which was not complex enough to capture the patterns in our data so its accuracy wasn't as good as thiers.

BONUS:

We have created and train another neural network model with 2 hidden layers and 4 hidden layers.

The 2 hidden layers model, will have the following specifications:

- It will consist of 2 Hidden layers.
- The first layer will include 256 nodes.
- The second layer will have 128 nodes.
- The third and final layer will contain 512 nodes.
- There will be 10 outputs.
- All layers will utilize the Relu activation function.

The 4 hidden layers model, will have the following specifications:

- It will consist of 4 Hidden layers.
- The first layer will include 512 nodes.
- The second layer will have 256 nodes.
- The third layer will contain 128 nodes.
- The fourth and final layer will contain 64 nodes.
- There will be 10 outputs.
- All layers will utilize the Relu activation function.

Training:

The 2 models will go through the same training as the initial model 4 stages of 10 epochs.

2 Hidden layers model:

Same as the initial model the accuracy stopped increasing after 25 epochs and stopped at 50.5% which is a good accuracy for such an architecture without such a complexity.

Here is the loss plotted against every epoch trained figure 19:

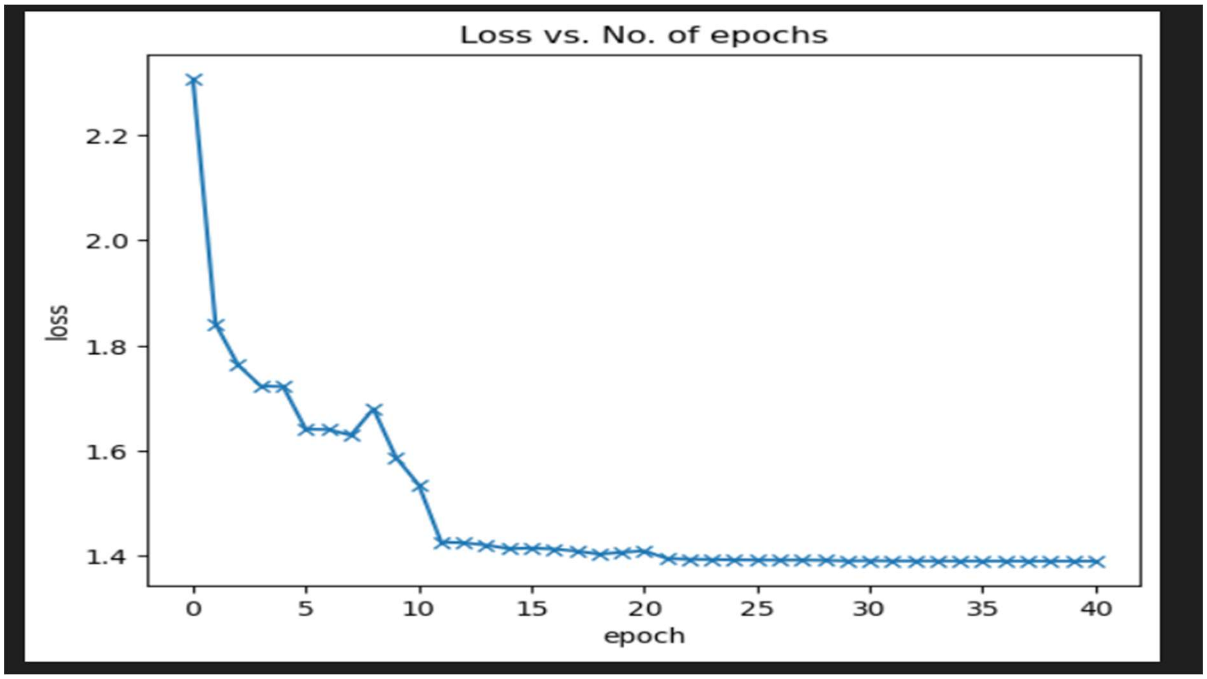
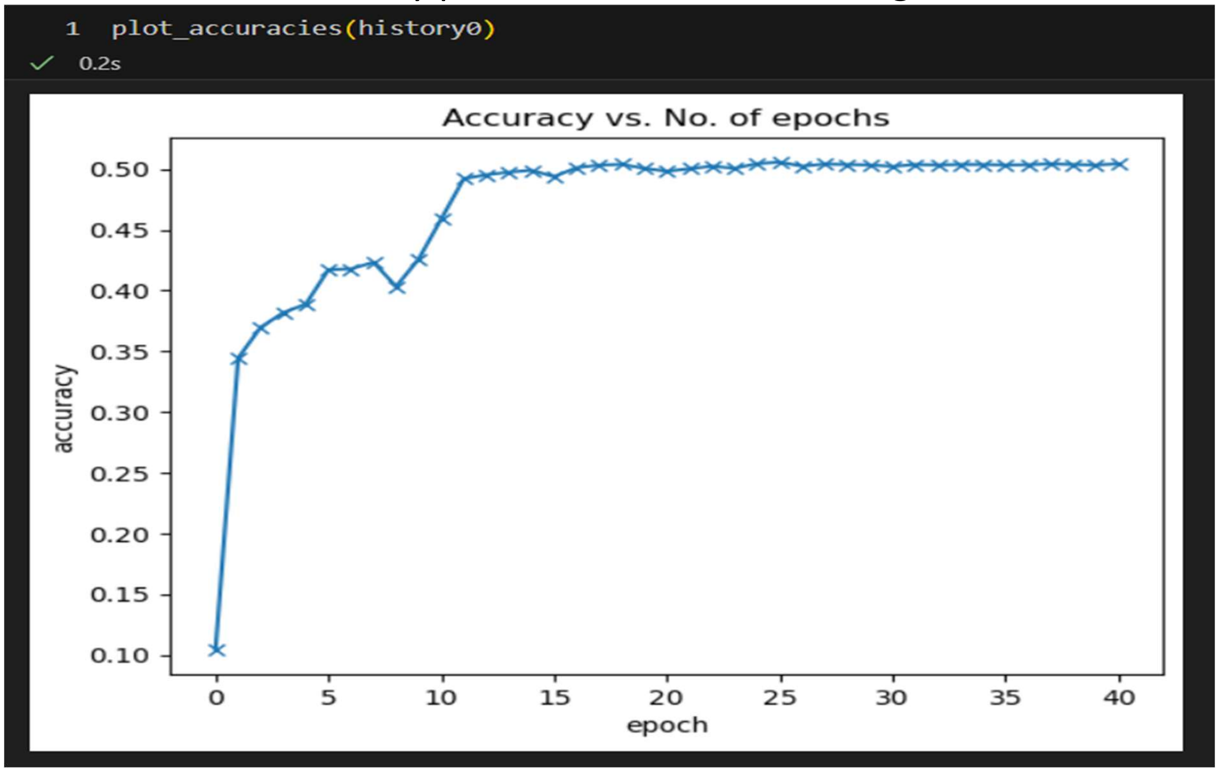


Figure 19

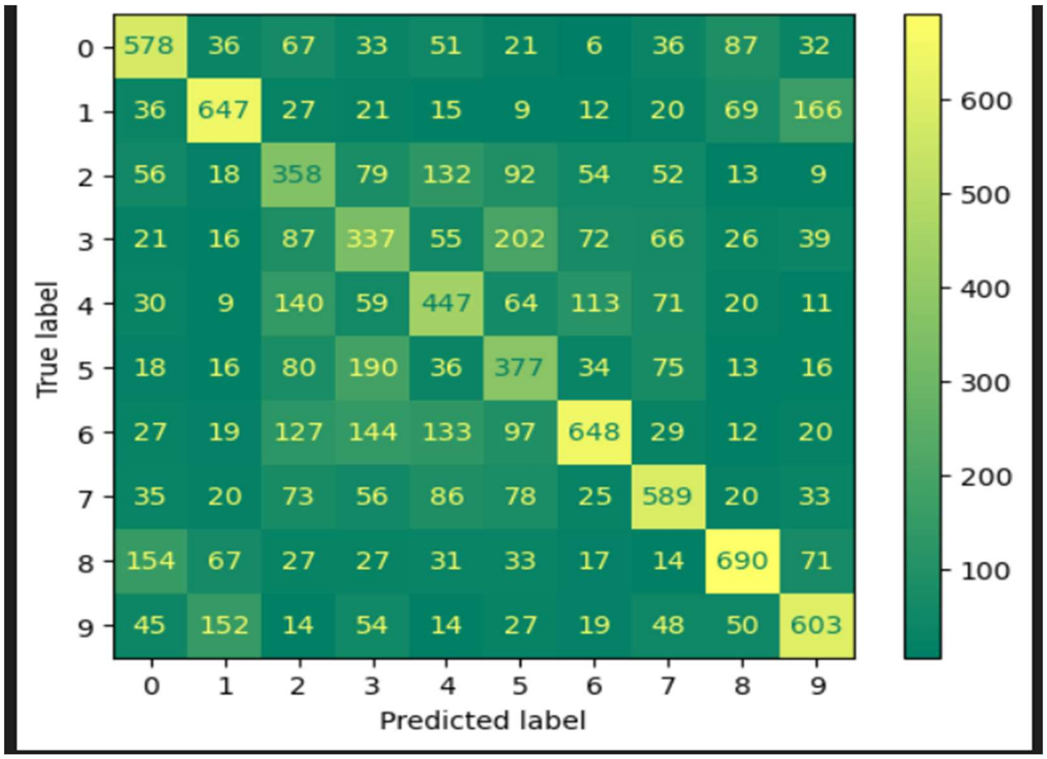
Here is the test accuracy plotted at the same rate figure 20:



```
plot_accuracies(history0)
```

Figure 20

And here is the confusion matrix figure 21:



True label \ Predicted label	0	1	2	3	4	5	6	7	8	9
0	578	36	67	33	51	21	6	36	87	32
1	36	647	27	21	15	9	12	20	69	166
2	56	18	358	79	132	92	54	52	13	9
3	21	16	87	337	55	202	72	66	26	39
4	30	9	140	59	447	64	113	71	20	11
5	18	16	80	190	36	377	34	75	13	16
6	27	19	127	144	133	97	648	29	12	20
7	35	20	73	56	86	78	25	589	20	33
8	154	67	27	27	31	33	17	14	690	71
9	45	152	14	54	14	27	19	48	50	603

True label
Predicted label
600
500
400
300
200
100

Figure 21

Here is a visualization of the image before and after training figure 22:

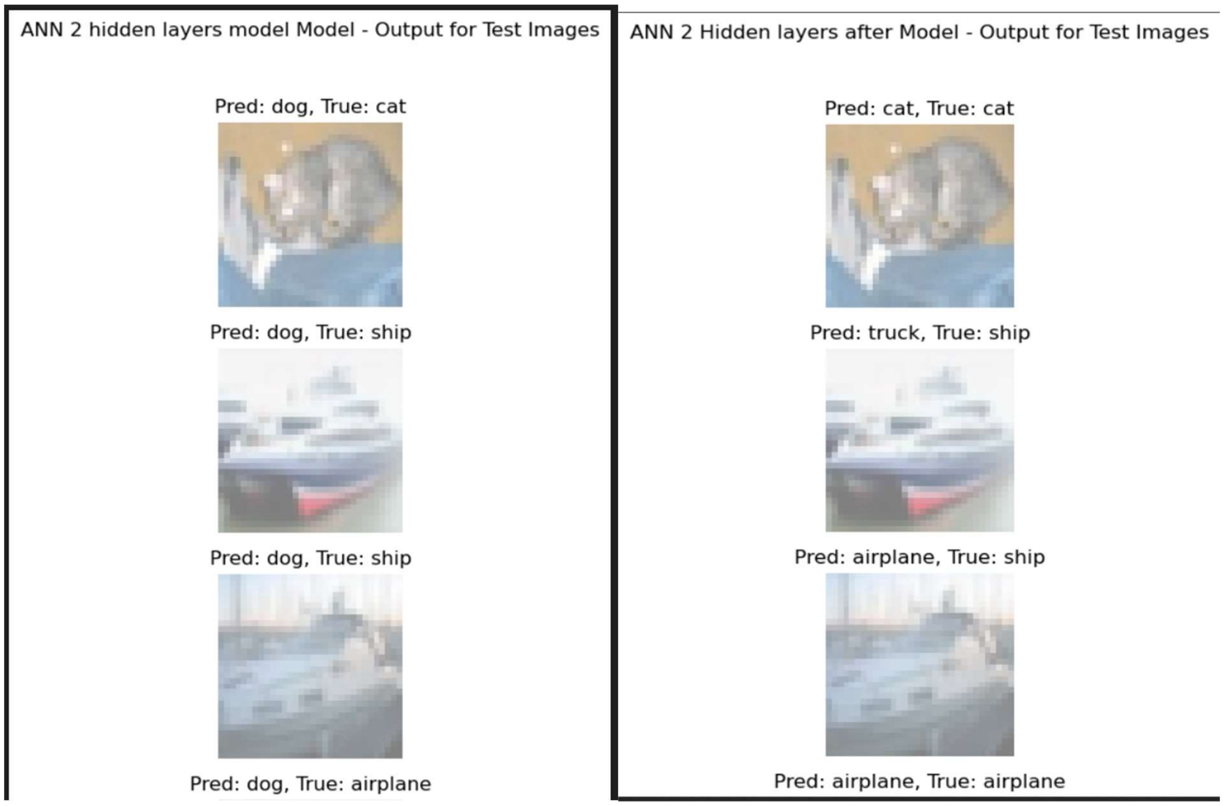

ANN 2 hidden layers model Model - Output for Test Images
Pred: dog, True: cat
Pred: dog, True: ship
Pred: dog, True: ship
Pred: dog, True: airplane
ANN 2 Hidden layers after Model - Output for Test Images
Pred: cat, True: cat
Pred: truck, True: ship
Pred: airplane, True: ship
Pred: airplane, True: airplane

Figure 22

4 Hidden layers model:
Same as the initial model the accuracy stopped increasing after 25 epochs and stopped at 50.5% which is a good accuracy for such an architecture without such a complexity.
Here is the loss plotted against every epoch trained figure 23:

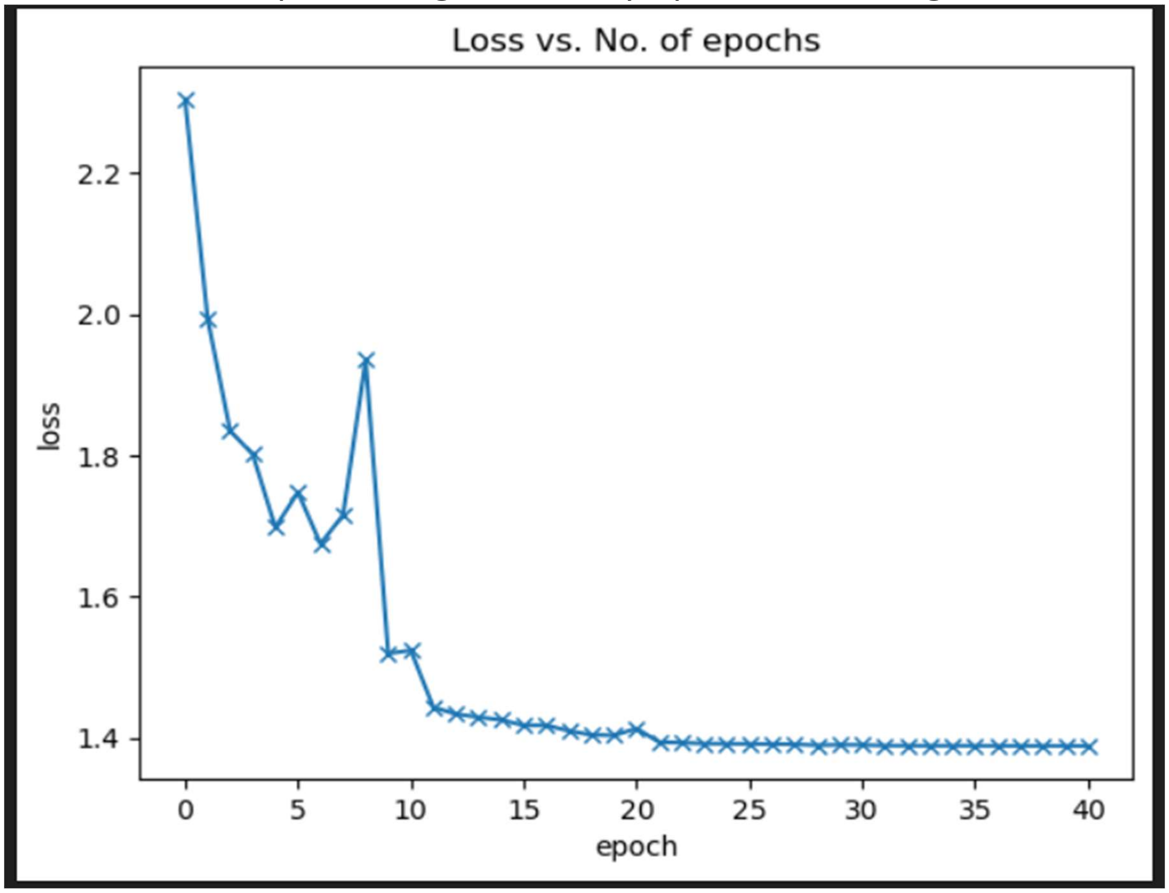


Figure 23

Here is the test accuracy plotted at the same rate figure 24:

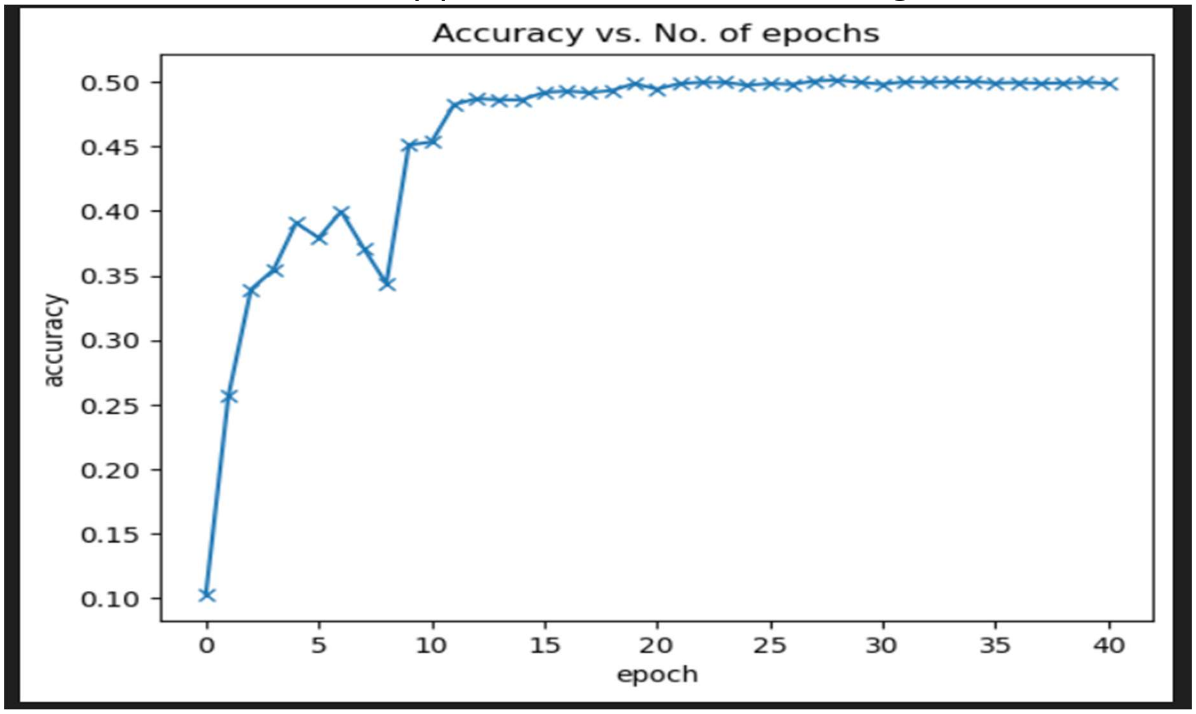


Figure 24

And here is the confusion matrix figure 25:

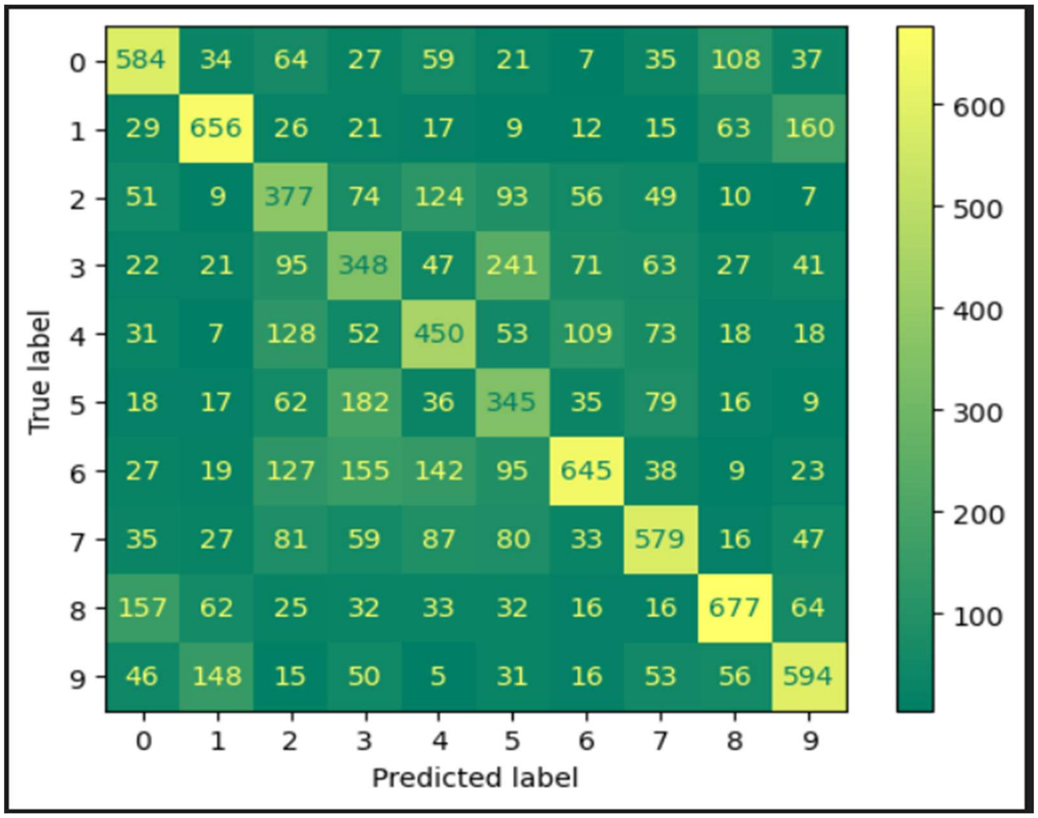


Figure 25

Here is a visualization of the image before and after training figure 26:

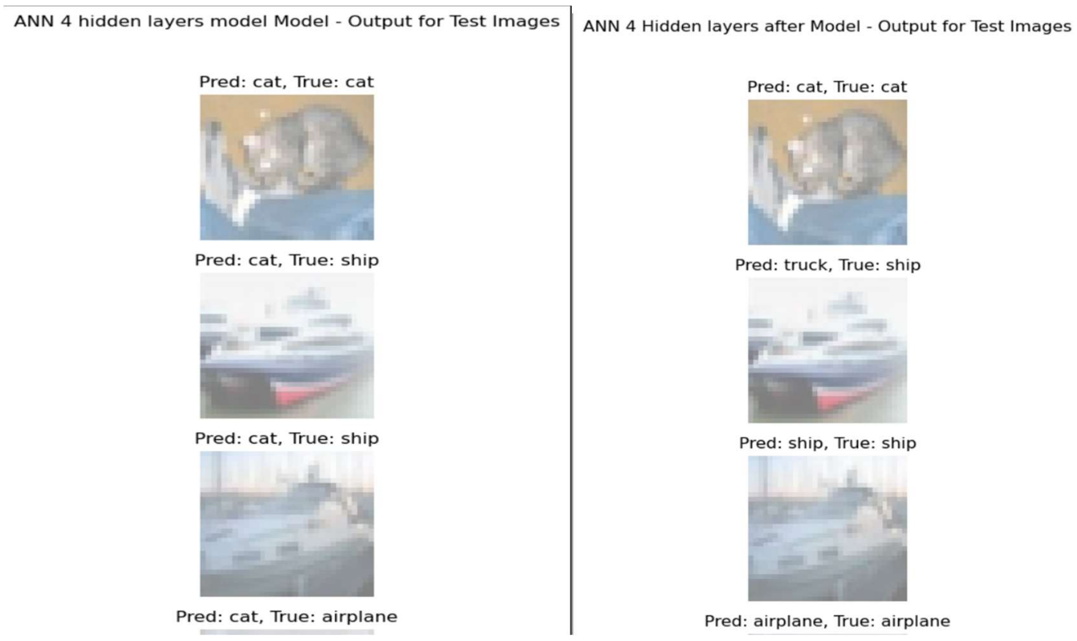

ANN 4 hidden layers model Model - Output for Test Images
Pred: cat, True: cat
Pred: cat, True: ship
Pred: cat, True: ship
Pred: cat, True: airplane
ANN 4 Hidden layers after Model - Output for Test Images
Pred: cat, True: cat
Pred: truck, True: ship
Pred: ship, True: ship
Pred: airplane, True: airplane

Figure 26