Project Machine Learning
Task 1: Deep Neural Networks
9 group

Tarelkin Evgenii
Steba Oxana

2021

.

1 Task 1

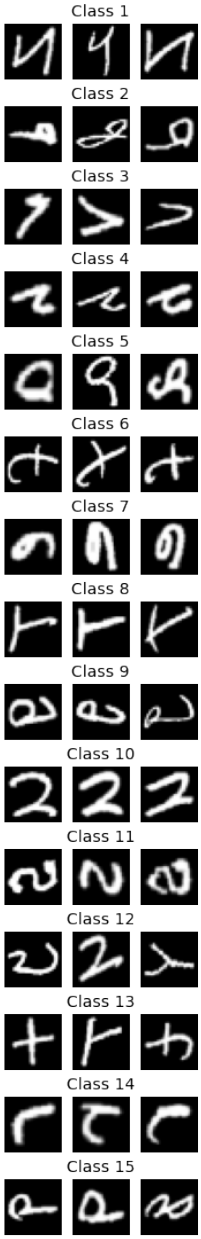

Figure 1: The set of three images from each class

a. Figure 1 shows three pictures from each class of the Strange Symbols dataset. We analyzed several pictures from each class and concluded that the symbols are similar to handwritten English letters. Also, these images are rotated a few degrees, and some of them are mirrored.
For example: Class 1: the letter "z" rotated 90 degrees to the right;
Class 2: the letter "d" rotated 90 degrees to the right and mirrored;
Class 3: the letter "v" rotated 90 degrees to the right;
Class 4: the letter "h" rotated 90 degrees to the right and mirrored;
Class 5: the capital letter "Q";
Class 6: the letter "t" rotated 90 degrees to the left;
Class 7: the letter "e" rotated 90 degrees to the right and mirrored;
Class 8: the capital letter "T" rotated 90 degrees to the right;
Class 9: the letter "g" rotated 90 degrees to the right and mirrored;
Class 10: the letter "u" rotated 90 degrees to the right and mirrored;
Class 11: the letter "s" rotated 90 degrees to the right and mirrored;
Class 12: the letter "y" rotated 90 degrees to the right and mirrored;
Class 13: the letter "f" rotated 90 degrees to the left;
Class 14: the letter "r" rotated 90 degrees to the right and mirrored;
Class 15: the letter "q" rotated 90 degrees to the right and mirrored;

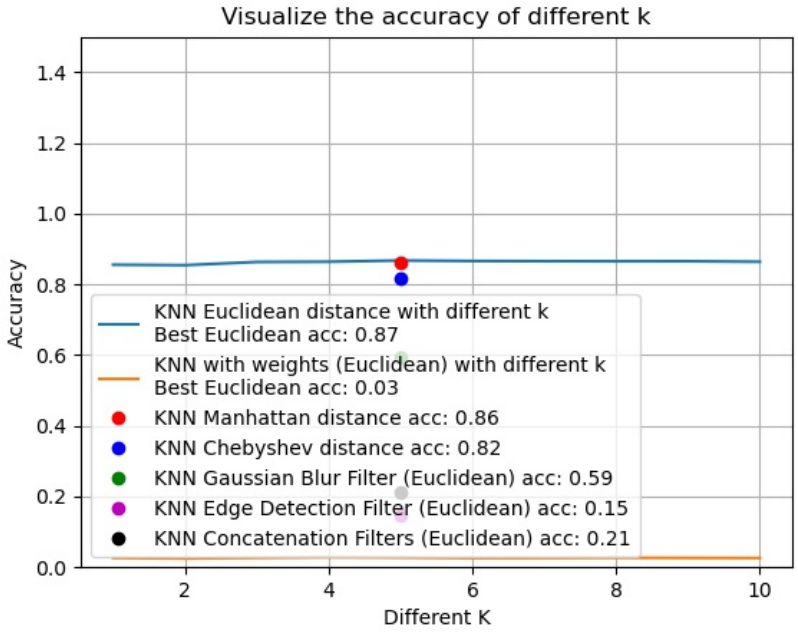

Figure 2: The result of the first task

c. Evaluated the performance of KNN on the Strange Symbols dataset using 5-fold cross-validation for all k and the Euclidean distance function. The result can be seen in Figure 2 as a plot and in the Table 1.

k	1	2	3	4	5	6	7	8	9	10
Accuracy	0.855	0.854	0.863	0.863	0.867	0.865	0.865	0.864	0.865	0.863

Table 1: Accuracy at various K-values in the KNN model Euclidean distance.

d. The blue line in Figure 2 is the KNN results using Euclidean distance with different k. The plot shows that the maximum accuracy of 87% is achieved at k=5. An accuracy value of 87% is a good estimate of the performance of the invisible data classifier. Considering that our dataset has 15 classes, the model accuracy of 87% is a good accuracy.

e. Two metrics are selected to measure the distance between two images from the Strange Symbols dataset:

1. Chebyshev distance. In this metric, the distance between two points is defined as the maximum value of the absolute value of the difference in coordinate values. Formula for calculating Chebyshev distance:

$$d_{xy} = \max_i (\mid x_i - y_i \mid)$$

2. Manhattan distance. In this metric, the distance between two points is measured along axes at right angles. Formula for calculating Manhattan distance:

$$d(x, y) = \sum_i \mid x_i - y_i \mid$$

For each distance, as well as for the Euclidean distance, a 5-fold cross-validation is applied. The results are shown in Figure 2 for k = 5:
Red mark - The accuracy of the KNN model with the Manhattan distance is 86%.
Green mark - The accuracy of the KNN model with the Chebyshev distance is 82%.
The result of Euclidean distance and Manhattan distance is much better than Chebyshev distance.

In the following tasks, we take the Euclidean distance k = 5 as the baseline.

g. Representations of functions for images are obtained using a convolutional operation. To complete the task, two filters are selected: blur and detect edge, Euclidean distance. When applying the KNN algorithm to features separately and to features together, the following results were obtained (Figure 2):
Green mark - KNN with blur filter. The accuracy is 59%.
Purple mark - KNN with edge detection filter. The accuracy is 15%.
Black mark - KNN with concatenation filter. The accuracy is 21%.
So far, our model with edge detection filter has the worst accuracy(15%).

h. To extend the KNN algorithm, each neighbor was assigned a weight inversely proportional to the distance from a given control point. To compare the results, 5-fold cross-validation was performed using Euclidean distance. The result is shown in Figure 2 and in the Table 2. The orange line is our weighted model. The best accuracy is 3% with k = 5.

k	1	2	3	4	5	6	7	8	9	10
Accuracy	0.026	0.024	0.026	0.028	0.027	0.026	0.025	0.026	0.026	0.026

Table 2: Accuracy at various K values in the KNN model with weights.

i. For KNN with Chebyshev distance (Figure 3),
KNN with Euclidean distance (Figure 4),
KNN with Manhattan distance (Figure 5),
KNN with Euclidean distance with blur filter (Figure 6),
KNN with Euclidean distance with edge detection filter (Figure 7),
KNN with Euclidean distance with concatenation filters (Figure 8) images are

displayed for incorrectly classified samples.
Predicted label - the predicted class of the model.
The "GT" above the images is the true class.
The first image is an image that is misclassified.
Next to it are 5 images - these are samples of images of the nearest five neighbors.

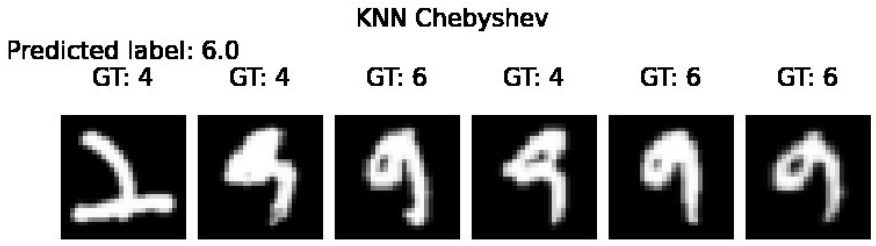

Figure 3: KNN Chebyshev distance. The ground truth along with the nearest neighbor image samples

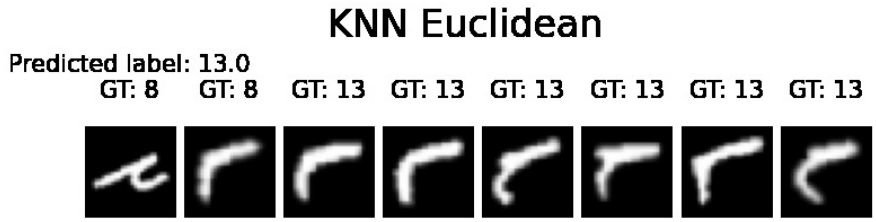

Figure 4: KNN Euclidean distance. The ground truth along with the nearest neighbor image samples

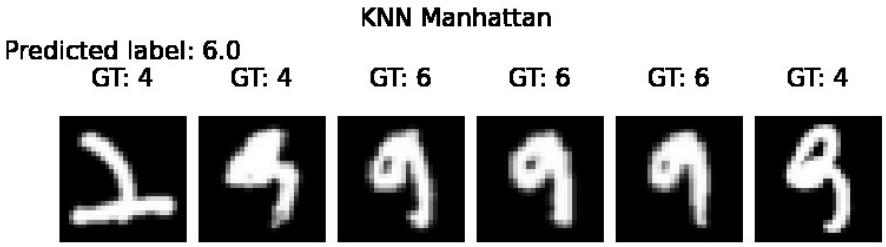

Figure 5: KNN Manhattan distance. The ground truth along with the nearest neighbor image samples

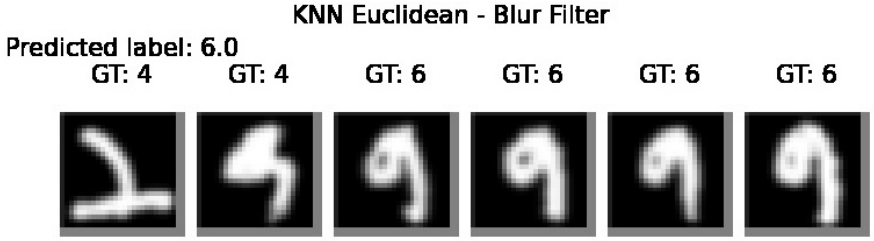

Figure 6: KNN Euclidean distance - Blur Filter. The ground truth along with the nearest neighbor image samples

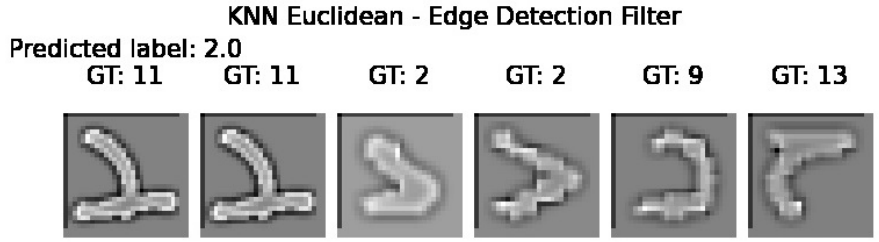

Figure 7: KNN Euclidean distance - Edge Detection Filter. The ground truth along with the nearest neighbor image samples

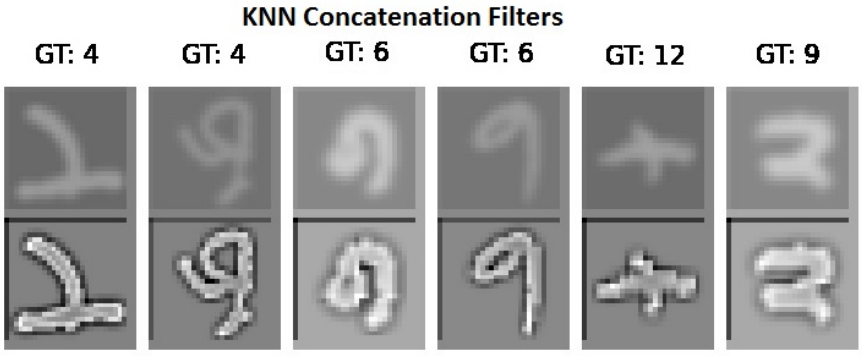

Figure 8: KNN Euclidean distance - Concatenation Filters. The ground truth along with the nearest neighbor image samples

j. In this task, we used the k-nearest neighbors (KNN) classification with different distance metrics.
Table 3 shows the classification accuracy for various distance metrics. For dis-

tance metrics (Chebyshev distance, Euclidean distance, Manhattan distance, Euclidean distance with weights), Euclidean distance provides the best performance.

Distance	Euclidean distance	Manhattan distance	Chebyshev distance	Euclidean distance with weights
Accuracy	0.87	0.86	0.82	0.03

Table 3: The KNN model accuracy for various distance metric

Filters	Blur	Edge detection	Concatenation
Accuracy	0.59	0.15	0.21

Table 4: Accuracy KNN models with different filters

We used 5-fold cross validation on the training set for all metrics. It was found that at K = 5 for all metrics, our KNN model shows the best accuracy. Tables 3 and 4 show that our model shows the best accuracy when using Euclidean distance.

2 Task 2

a. Activation function. The activation function defines the output signal, which is determined by the input signal (set of input signals). It is one of the most important aspects of a deep neural network. The purpose of the activation function is to give the neural network non-linear factors so that the neural network can solve more complex problems, as well as increase the neural network's ability to learn. Frequently used activation functions: binary step function, sigmoid, ReLU, linear, etc.

Loss function. The loss function describes how far the result is from the expected result. It indicates the amount of error made by the model when predicting it. The loss function will take two elements as input: the output of the model and the expected true value.Frequently used loss functions: Maximum likelihood estimation, the cross-entropy loss function, etc.

Hyperparameters. Hyperparameters are parameters of algorithms, the values of which are set before starting the learning process. For example, the number of hidden layers and neurons in the activation function, the coefficient of the learning rate, the number of training iterations, the error rate, etc.

Optimizer. An optimizer is an algorithm that is used to achieve better results, such as helping to speed up learning. This method is used to change parameters such as weights and learning rate to make the model work correctly

and quickly. Also to reduce losses.

Epoch. The epoch is training a neural network with all the training data in one cycle. Epoch is one of the repetitive stages of training set processing. One epoch leads to underfitting, and an excess of epochs leads to overfitting.

Underfitting. Underfitting is a situation when a model cannot find a function that describes the data well and cannot generate new data. The underfitting model contains too few features. One of the most common reasons for underfitting is when the complexity of the data is greater than the complexity of the model. The solution to the problem is to complicate the model.

Overfitting. Overfitting is the opposite of underfitting, where the model is too complex and universal. Overfitting occurs when the model learns in detail the noise in the training data, so that noise or random fluctuations in the training data are perceived and learned by the model as a basic concept.

Training set. Training set is part of a dataset that is initially entered into the program for learning. The training sets are fed into the model's algorithm to teach them to make predictions or to do the required task.

Test set. Test set is part of a dataset that is used to test a machine learning program after it has been trained on the training dataset.

Validation set. Validation set is part of the dataset, the basis for checking the model. This data is used to assess the skill of the model when tuning the hyperparameters of the model.

c. We use two different ways to create the deep neural network. The first one is using the nn. Sequential, and the second is using the nn.Module. Several experiments were carried out, and results are presented in the Table 5.

1. The first experiment with the initial parameters: Architecture: 3 fully connected layers(512, 256, 15 neurons) with the ReLU activation function
Loss: Cross Entropy Loss
Learning rate: 0.01
Optimizer: Stochastic Gradient Descent
Epochs: 50
As we can see from the Table, the results are very close to each other. In both cases accuracy is more than 90%, accuracy is 0.92187 and 0.91406 in the Sequential and the Module, respectively. The Module takes less time to train the model than the Sequential, 5.539 mins opposite to the 6.313 mins. (Figure 9)

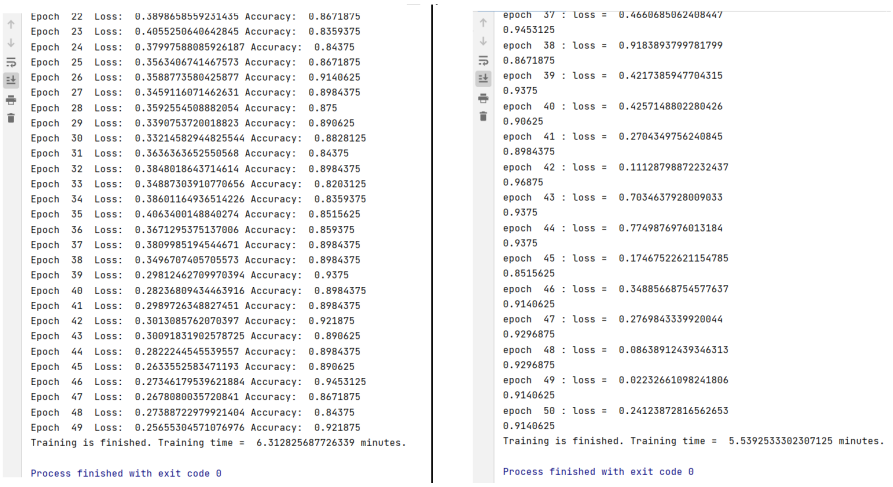

Figure 9: Accuracy and loss of the Sequential and Module models, respectively, trained with initial parameters.

2. We increase the number of epochs

Epochs: 60

Accuracy is 0.89 in the Sequential. Accuracy is 0.86 in the Module. But training time is very long (20 and 35 minutes).(Figure 10)

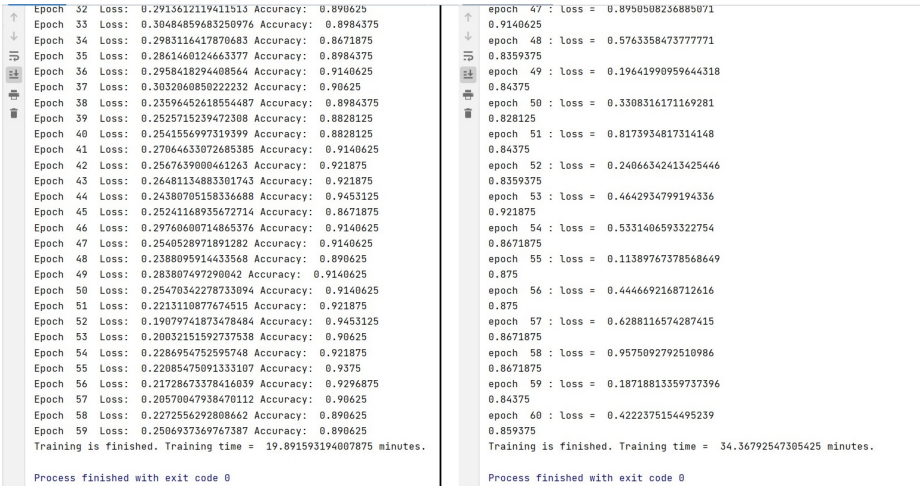

Figure 10: Training information of the models with the increased number of epochs during the second experiment. On the left image is the Sequential model, on the right image is the Module model.

3. The third experiment. Changed parameters:

Optimizer: Adagrad
In this experiment model created using the nn.Sequential performs better than the model created by the nn.Module but takes more time (6.206min – Sequential and 5.884min - Module). Accuracy is 0.14844 in Sequential, and 0.04688 in Module. These accuracies are very low for the model. And looking in the Figures, we can say that models are not trained, because the accuracy and the loss are equal all of the training time(Figure 11).

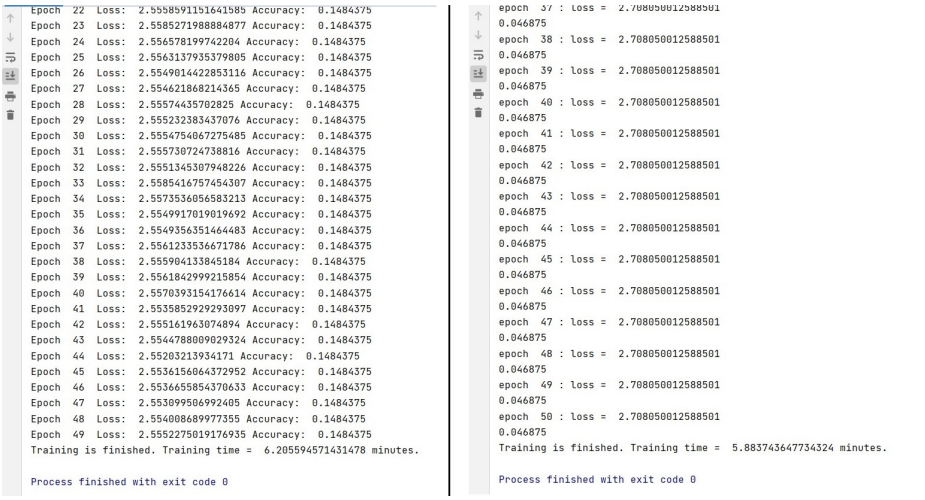

Figure 11: Training information of the Sequential model and Module model, respectively, with the Adagrad optimizer.

4. In the fourth experiment, two elements are changed:
Optimizer: Adam
Learning rate: 0.05
In this experiment, none of the models show a good result. The accuracy is lower than 10%, and training time is almost 6 minutes. And as in the previous experiment, the accuracy and loss are not changed during the training of the model (Figure 12)

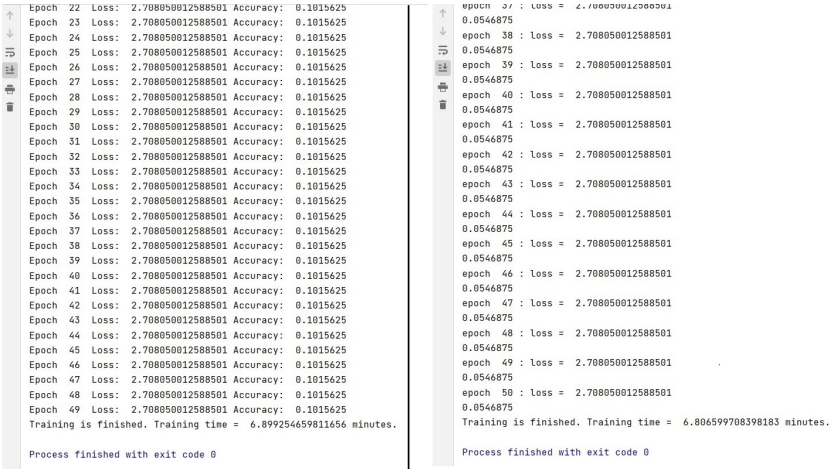

Figure 12: Training information of the Sequential model and Module model, respectively, with the Adam optimizer and learning rate 0.05.

5. We change the architecture of the model:

Architecture: 4 fully connected layers(1024, 512, 256, 15 neurons) with the ReLU activation function

The accuracy of the two models is increased and become more than 95%. Accuracy of two models is equal to 0.95312. Training time is almost equal too (8.538 min – Sequential, 8.727 min - Module) (Figure 13).

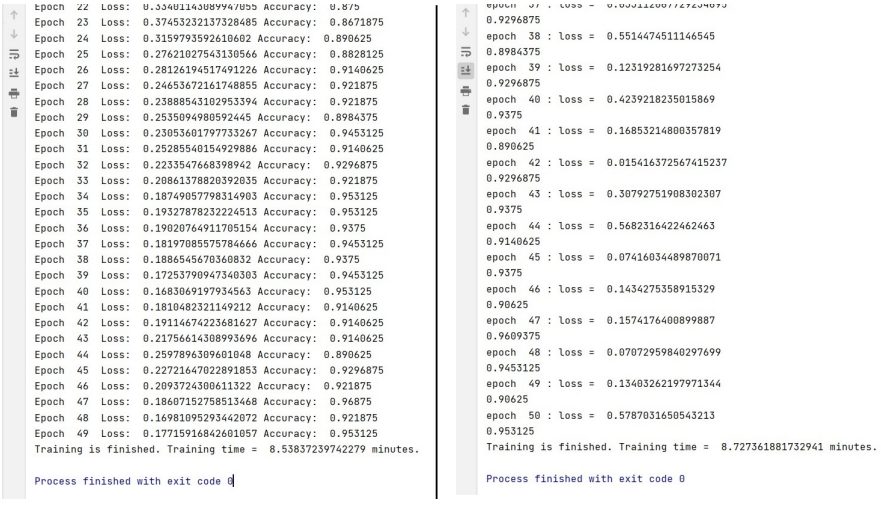

Figure 13: Training information of the models with the increased number of layers. On the left image is the Sequential model, on the right image is the Module model.

6. Using the previous architecture we change the learning rate:
Learning rate: 0.02
The increased learning rate decreases the accuracy of the models and training time. As we can see from the Table, accuracy from 0.95 falls to 0.59 in Sequential, accuracy of Module decreased from 0.95 to 0.5. But in the some middle epochs accuracy was greater. For example, in 34 epoch in the Figure accuracy was 0.71 in Sequential; in 38 epoch in the Figure accuracy was 0.77 in Module. (Figure 14)

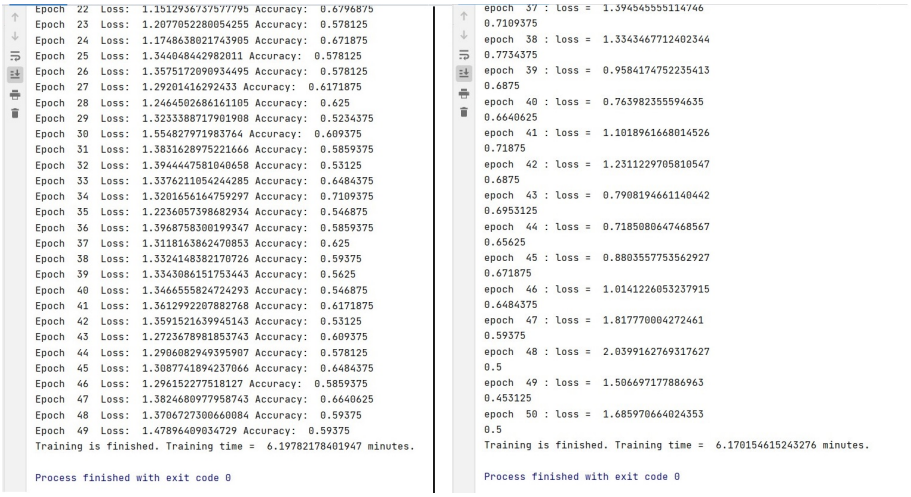

Figure 14: Accuracy, loss, and training time of Sequential and Module models with the increased number of layers and learning rate 0.02.

7. We use the previous architecture with the four fully connected layers.
Optimizer: Adam
Learning rate: 0.03
Accuracy is very low in this experiment, less than 10%. The accuracy and the loss are constant during the model training (Figure 15).

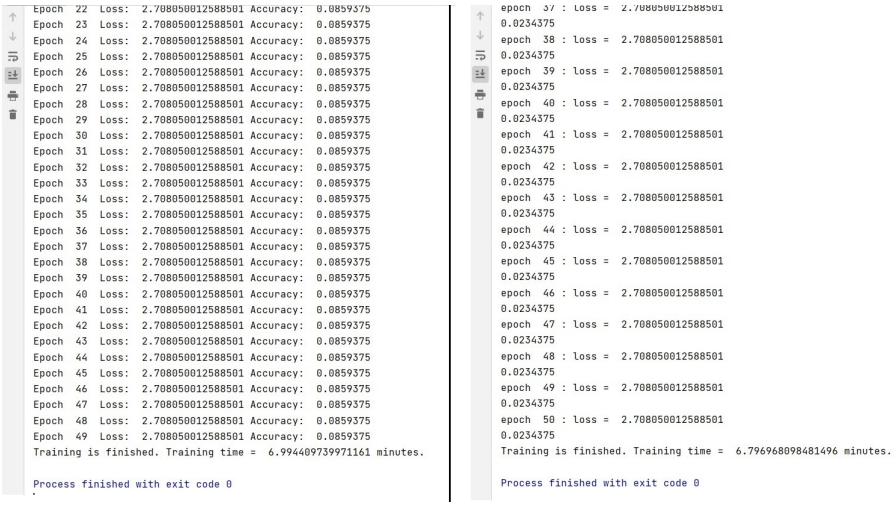

Figure 15: Accuracy, loss, and training time of Sequential and Module models with the increased number of layers, Adam optimizer, and learning rate 0.03.

No of experiment	nn.Sequential			nn.Module		
	acc	loss	time	acc	loss	time
1	0.92187	0.25655	6.313 mins	0.91406	0.24123	5.539 mins
2	0.89063	0.25069	19.892 mins	0.85938	0.42224	34.368 mins
3	0.14844	2.55523	6.206 mins	0.04688	2.70805	5.884 mins
4	0.10156	2.70805	6.9 mins	0.05469	2.70805	6.8 mins
5	0.95312	0.17716	8.538 mins	0.95312	0.57870	8.727 mins
6	0.59375	1.47896	6.198 mins	0.5	1.68597	6.170 mins
7	0.08594	2.70805	6.994 mins	0.02344	2.70805	6.797 mins

Table 5: Experiment results

As we can see from the Table 5, the fastest performance of Sequential is the experiment with four fully connected layers and learning rate equal to 0.02 (6.198 min). The fastest performance of Module is the experiment with the initial parameters (5.539 min). The highest accuracy in the experiment with four fully connected layers – 0.95312. The lowest accuracy in the experiment with four fully connected layers, Adam optimizer and learning rate equals to

0.03.

We will use the architecture below as we needed to merge our architecture from 2b and add CNN layers on top. Accuracy approximately 92%

```
CNNModel(
  (cnn1): Conv2d(1, 16, kernel_size=(3, 3), stride=(1, 1))
  (relu1): ReLU()
  (maxpool1): MaxPool2d(kernel_size=2, stride=2, padding=0,
  dilation=1, ceil_mode=False)
  (cnn2): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1))
  (relu2): ReLU()
  (maxpool2): MaxPool2d(kernel_size=2, stride=2, padding=0,
  dilation=1, ceil_mode=False)
  (fc1): Linear(in_features=800, out_features=512, bias=True)
  (fc2): Linear(in_features=512, out_features=256, bias=True)
  (fc3): Linear(in_features=256, out_features=15, bias=True)
)
```

But when using the following architecture, as below, the accuracy reaches 97-98%. When we add additional FullyConnected layers to our model, we lose precision.

```
CNNModel(
  (cnn1): Conv2d(1, 16, kernel_size=(3, 3), stride=(1, 1))
  (relu1): ReLU()
  (maxpool1): MaxPool2d(kernel_size=2, stride=2, padding=0,
  dilation=1, ceil_mode=False)
  (cnn2): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1))
  (relu2): ReLU()
  (maxpool2): MaxPool2d(kernel_size=2, stride=2, padding=0,
  dilation=1, ceil_mode=False)
  (fc1): Linear(in_features=800, out_features=15, bias=True)
)
```

When we add additional FullyConnected layers to our model, we lose precision. The trained network on the Strange Symbols dataset is plotted that shows the number of epochs on the x-axis and the learning loss on the y-axis. Figures 16 and 17 depict the accuracy and loss of the neural network at each epoch, respectively.

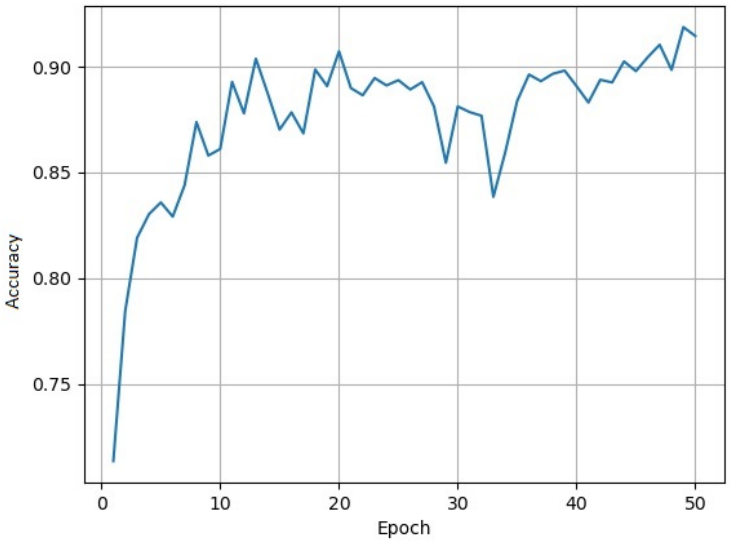

Figure 16: Accuracy of the trained network in each epoch

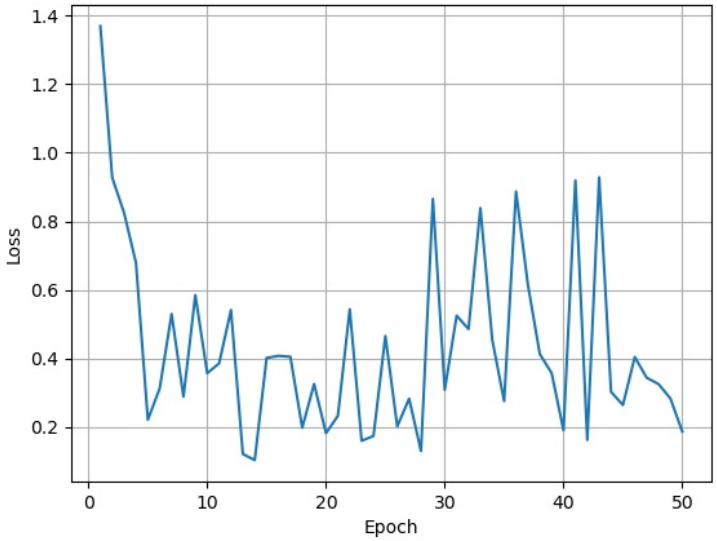

Figure 17: Loss of the trained network in each epoch

d. To better understand the behavior of the classifier, the inputs are computed and interpreted into a confusion matrix (Figure 18) for our classifier.

```
[[934   0   1   0   2   3   8  18  15   0   3   2  13   0   1]
 [  1 972   2   5   3   2   1   0   5   0   0   3   6   0   0]
 [  4   4 792   9   0   0   1   0   3  84   1  71   9  22   0]
 [  1   6   2 950   2   4   0   0   0   2   1  10  17   2   3]
 [ 10  19   0   4 846   4  56   0  32  10   1   0   2   5  11]
 [  2   1   0   3   0 937   3   0  12   0  11   1  25   1   4]
 [  2   0   1   0   0   5 967   0  13   0   2   0   7   0   3]
 [  1   0   1   0   1  11   0 896   0   0   3   1  83   2   1]
 [  0   6   1   3  34  13  10   0 758   0  20   3   4   1 147]
 [  6   7  61  13  21   0   1   0  16 858   0  12   4   0   1]
 [  0   0   1   0   3  16   2   0  21   0 952   0   3   1   1]
 [  0   6   6   1   1   2   1   4  17   4   3 929  14   4   8]
 [  3   0   0   1   1  52   3  20   1   0   1   3 915   0   0]
 [  0   3  18   0   1  18   5  23   7   9  11  17   9 879   0]
 [  2   2   1   3  38  12  10   0 227   4   3   1   3   1 693]]
```

Figure 18: Confusion matrix for our classifier

For each class, ten images were constructed, for which the network predicts the label correctly and most confidently:

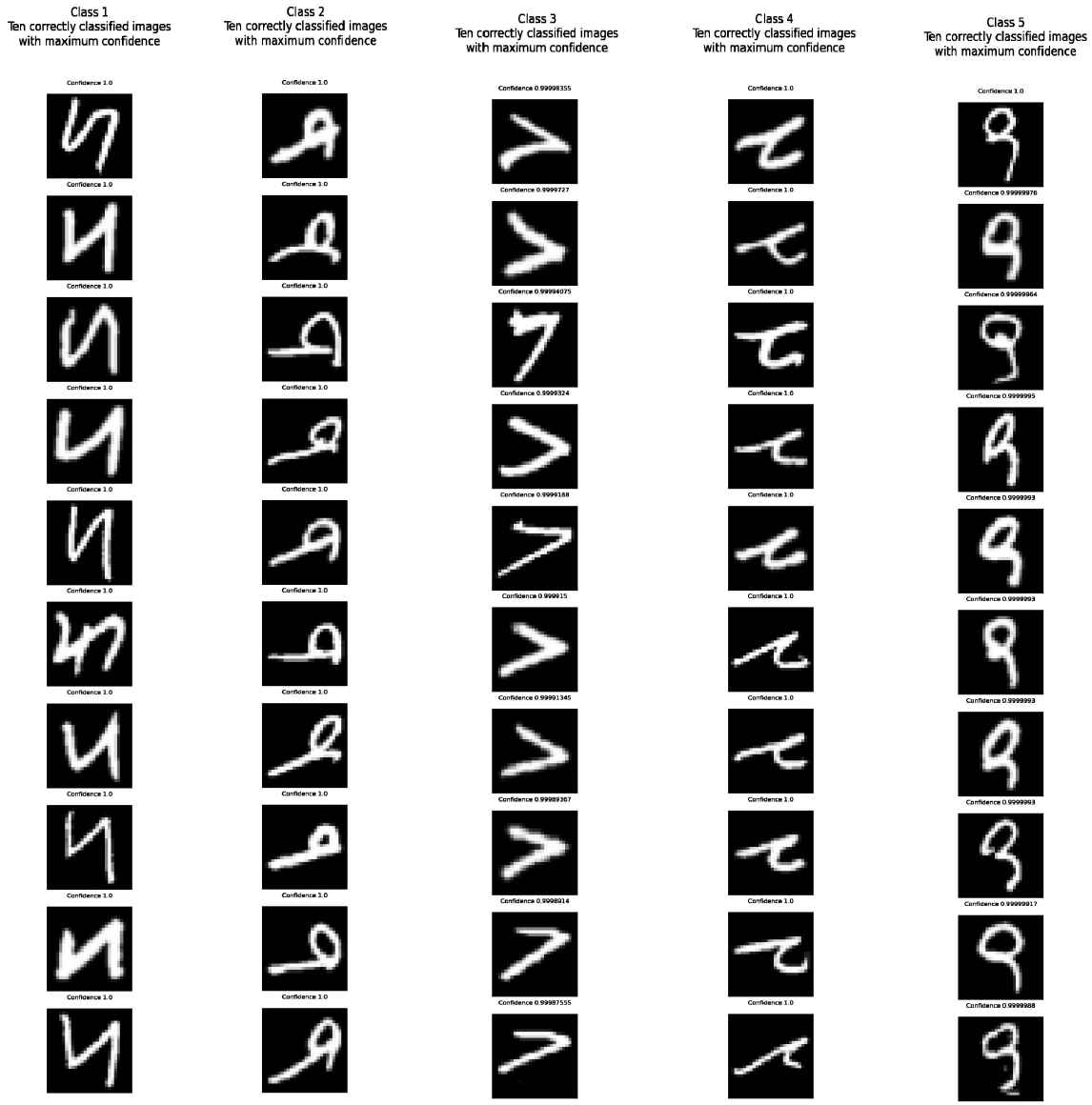

Figure 19: Ten correctly classified images with maximum confidence for 1-5 classes

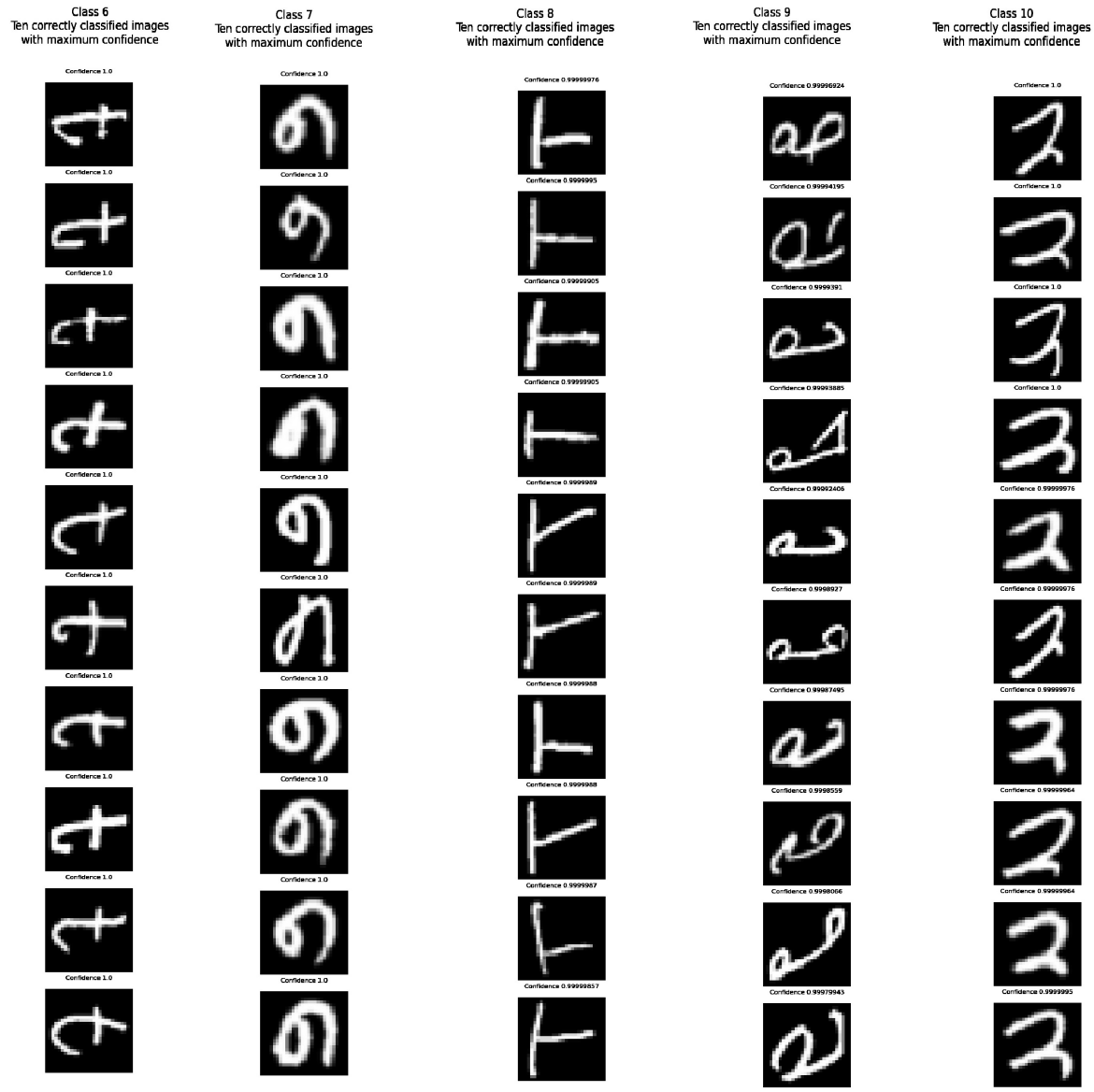

Figure 20: Ten correctly classified images with maximum confidence for 6-10 classes

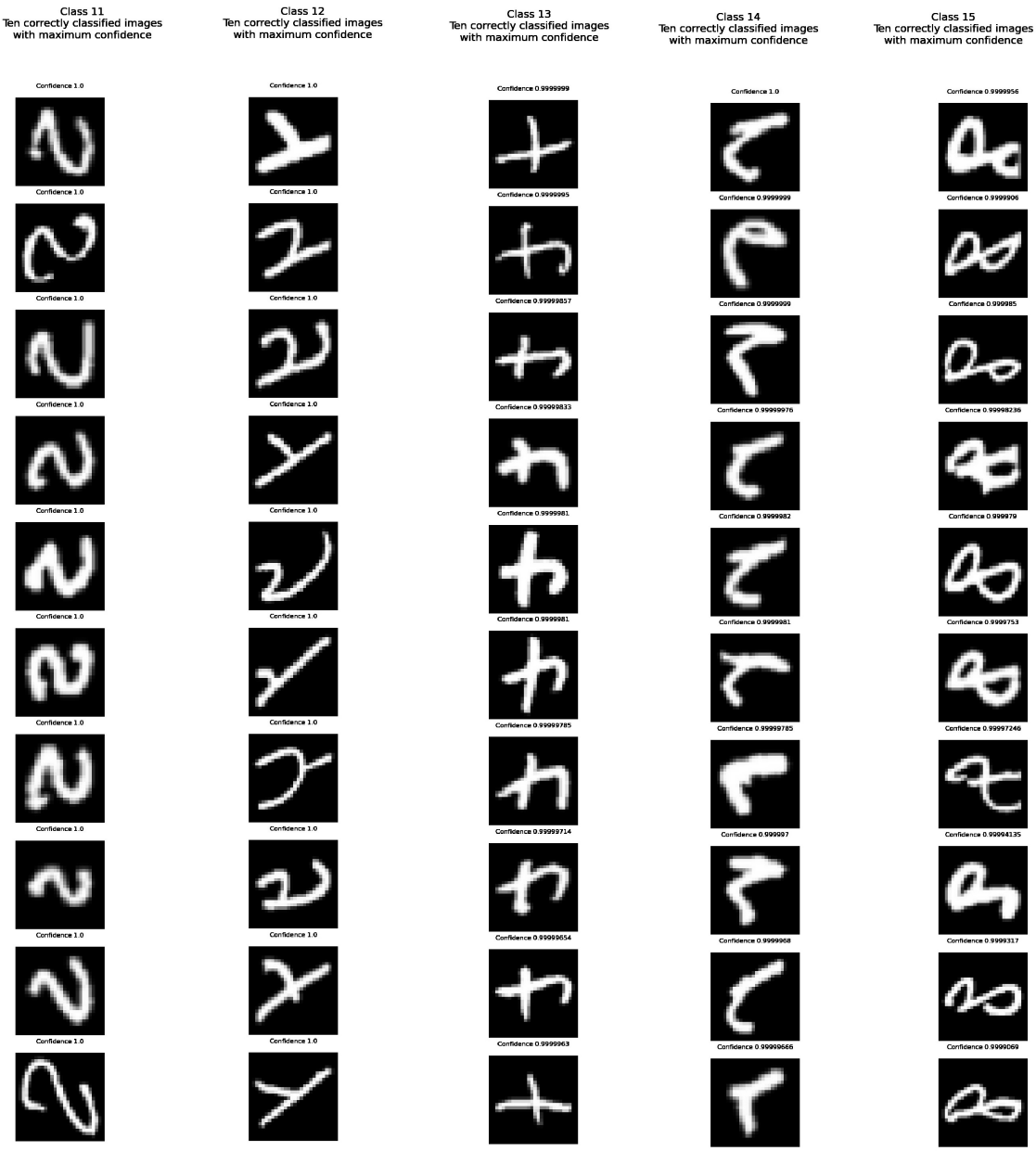

Figure 21: Ten correctly classified images with maximum confidence for 11-15 classes

f. Implemented 2D Batch Normalization layer in PyTorch. Below is a con-

fusion matrix (Figure 22) and two graphs with accuracy (Figure 23) and loss (Figure 24) at each epoch.

```
[[984   1   0   1   0   0   5   1   2   0   0   3   3   0   0]
 [  0 971   6   1  12   1   1   0   0   2   0   6   0   0   0]
 [  1   2 877   5   2   0   0   0   0  65   0  31   3  14   0]
 [  1   1   0 987   0   0   0   0   0   3   0   5   3   0   0]
 [ 12  12   0   7 919   1   2   0   9   8   0   0   0   0  30]
 [  2   3   1  11   0 894   6   2   5   0   2   9  53   3   9]
 [  0   0   1   2  31   2 934   0   2   8   0   0   0   2  18]
 [ 11   0   0   0   1   0   0 918   1   0   0   1  56  11   1]
 [  6  10   1   3  35   9   4   2 488   2   1  23   3   4 409]
 [  2   3  22   9   6   0   0   0   0 947   0   8   1   0   2]
 [  0   2   0   9   3  13   1   0  53   2 884   7   5   2  19]
 [  5   0  28   2   0   0   0   0   1   3   0 951   4   4   2]
 [ 10   2   2  12   0   2   4  33   0   0   0  20 911   1   3]
 [  4   1  16   4  12   7   3   5   0   1   0  15  26 904   2]
 [  1   0   2  12  30   1   1   2  40   3   0   6   0   1 901]]
```

Figure 22: Confusion matrix for our model with 2D Batch Normalization

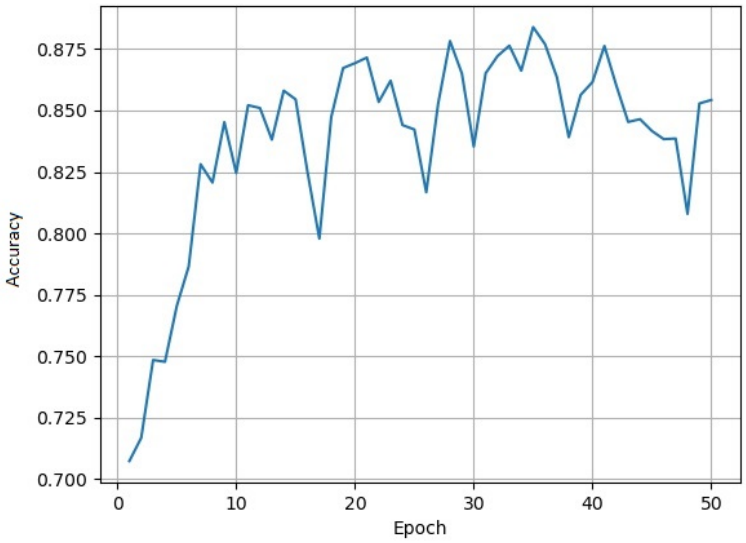

Figure 23: Accuracy of the trained network in each epoch in our model with 2D Batch Normalization

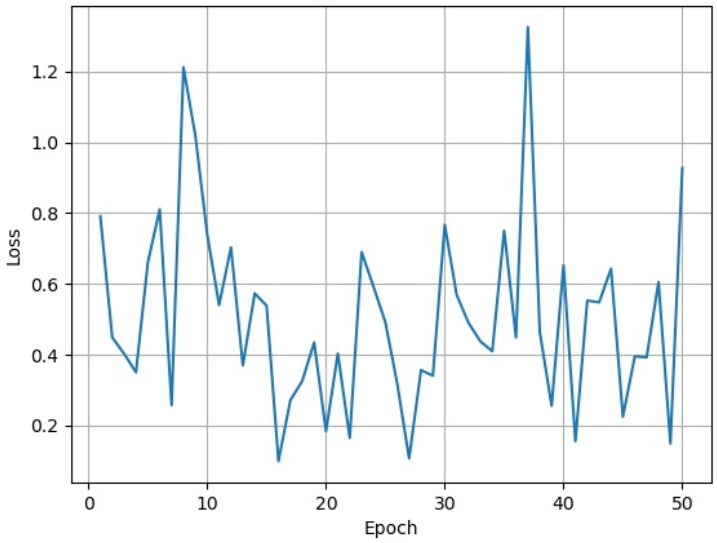

Figure 24: Loss of the trained network in each epoch in our model with 2D Batch Normalization

According to the theory, 2D batch normalization should normalize data in addition to simple normalization of dataset data, which makes it possible to align this data on each of the layers separately and in the output as a whole. That is, the data should ultimately be more ordered. However, in our case, 2D batch normalization did not perform the expected data ordering.

According to the plots (Figure 16 and 23) , the accuracy in the CNN model reaches 93% (task 2.1.c), but CNN model with 2D batch normalization it does not even reach 90%

According to the graphs (figure 17 and 24)), the loss in the CNN model reaches 0.2 (2.1.c), and in the CNN model with 2D packet normalization, the loss is more than 0.9.

According to confusion matrices(figure 18 and 22), recognition of 9 classes improved after 2D batch normalization. For example, it correctly recognizes 984 Grade 1 images, while 934 images were correctly recognized by CNN.

A significant improvement can be seen in the recognition of the 15th class, it now correctly recognizes 901 images instead of 693. The 9th grade became worse to be recognized, with CNN it correctly recognizing 758 images, and after batch 2D normalization, only 488 and 409 images from the 9th grade were recognized as the 15th grade.