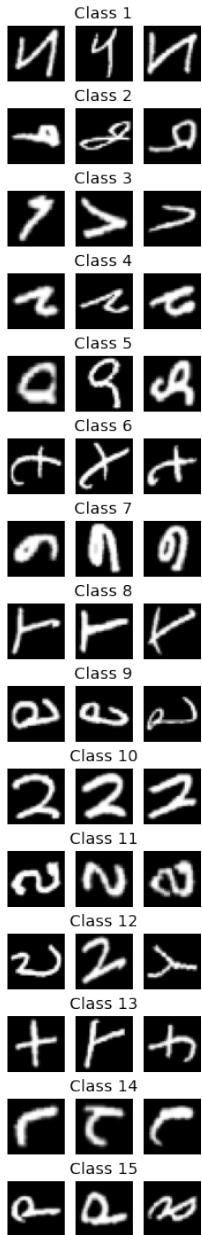


Project Machine Learning
Task 1: Deep Neural Networks
9 group

Tarelkin Evgenii
Steba Oxana

2021

1 Task 1



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 1: The set of three images from each class

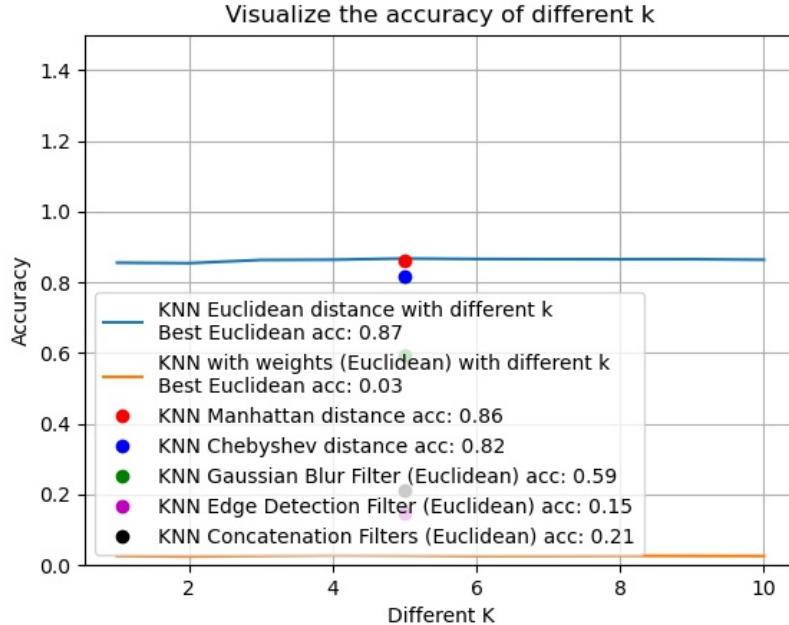


Figure 2: The result of the first task

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:

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

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

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

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.

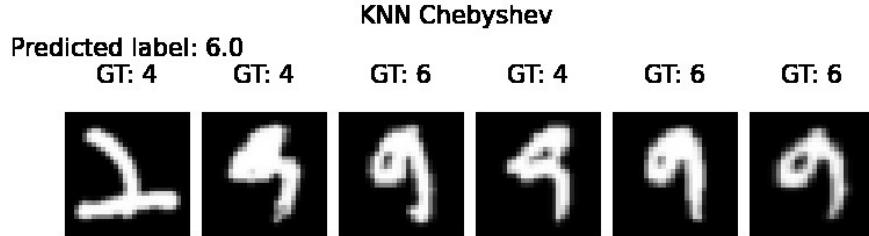


Figure 3: KNN Chebyshev distance. The ground truth along with the nearest neighbor image samples

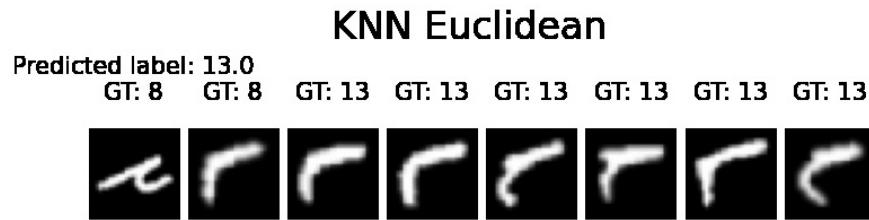


Figure 4: KNN Euclidean distance. The ground truth along with the nearest neighbor image samples

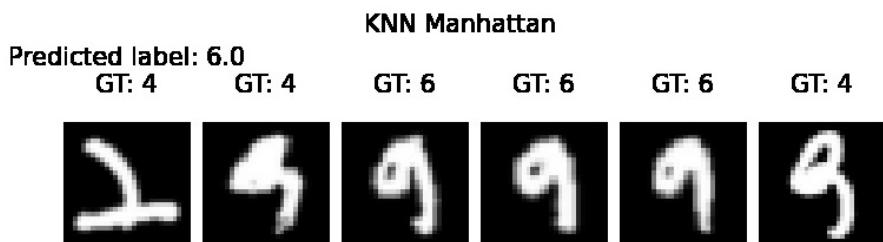


Figure 5: KNN Manhattan distance. The ground truth along with the nearest neighbor image samples

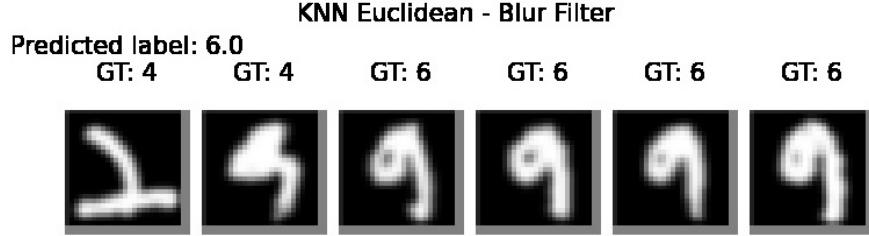


Figure 6: KNN Euclidean distance - Blur Filter. The ground truth along with the nearest neighbor image samples

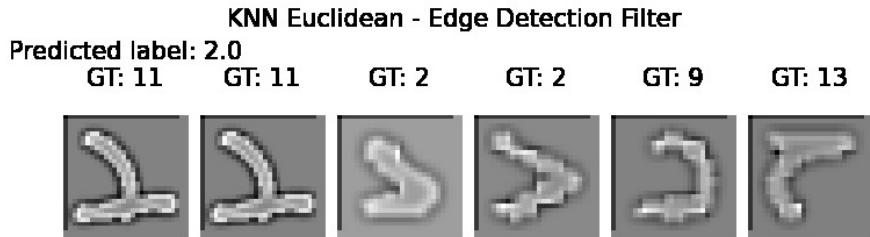


Figure 7: KNN Euclidean distance - Edge Detection Filter. The ground truth along with the nearest neighbor image samples

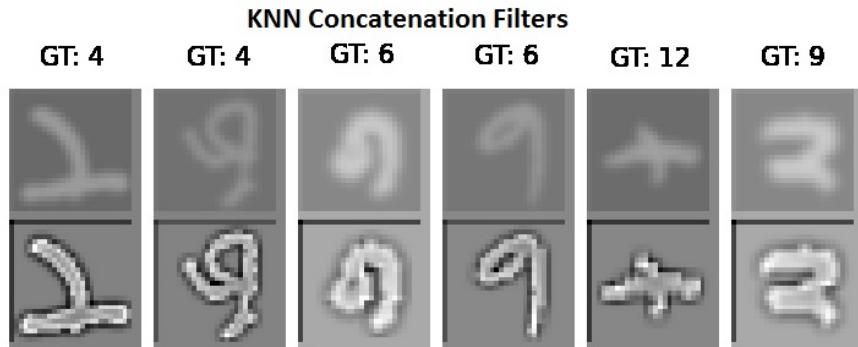


Figure 8: KNN Euclidean distance - Concatenation Filters. The ground truth along with the nearest neighbor image samples

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)

```

epoch 22 : Loss: 0.38986585923145 Accuracy: 0.8671875
Epoch 23 : Loss: 0.4055250640642845 Accuracy: 0.8359375
Epoch 24 : Loss: 0.37997588805926187 Accuracy: 0.84375
Epoch 25 : Loss: 0.356340671467573 Accuracy: 0.8671875
Epoch 26 : Loss: 0.3588773580425877 Accuracy: 0.9140625
Epoch 27 : Loss: 0.3459116071462631 Accuracy: 0.8984375
Epoch 28 : Loss: 0.3592554508882054 Accuracy: 0.875
Epoch 29 : Loss: 0.3390753720018823 Accuracy: 0.89625
Epoch 30 : Loss: 0.35214582944628544 Accuracy: 0.8828125
Epoch 31 : Loss: 0.36363636362550568 Accuracy: 0.84375
Epoch 32 : Loss: 0.3848018643714614 Accuracy: 0.8984375
Epoch 33 : Loss: 0.34887303910776659 Accuracy: 0.8203125
Epoch 34 : Loss: 0.38601164936514226 Accuracy: 0.8539375
Epoch 35 : Loss: 0.4063400148840274 Accuracy: 0.8515625
Epoch 36 : Loss: 0.3671295375137086 Accuracy: 0.859375
Epoch 37 : Loss: 0.3809985194544671 Accuracy: 0.8984375
Epoch 38 : Loss: 0.34967074605705573 Accuracy: 0.8984375
Epoch 39 : Loss: 0.2981246278970394 Accuracy: 0.9375
Epoch 40 : Loss: 0.28236809434645916 Accuracy: 0.8984375
Epoch 41 : Loss: 0.2989726348827451 Accuracy: 0.8984375
Epoch 42 : Loss: 0.3013085762070397 Accuracy: 0.921875
Epoch 43 : Loss: 0.30891831902578725 Accuracy: 0.89625
Epoch 44 : Loss: 0.2822244545395957 Accuracy: 0.8984375
Epoch 45 : Loss: 0.2633552583471193 Accuracy: 0.89625
Epoch 46 : Loss: 0.273461795396211884 Accuracy: 0.9453125
Epoch 47 : Loss: 0.2678080035720841 Accuracy: 0.8671875
Epoch 48 : Loss: 0.2738872297921404 Accuracy: 0.84375
Epoch 49 : Loss: 0.256553094571076976 Accuracy: 0.921875
Training is finished. Training time = 6.312825687726339 minutes.

Process finished with exit code 0
```

```

epoch 37 : Loss: 0.4660685962408847
0.9453125
epoch 38 : loss = 0.9183893799781799
0.8671875
epoch 39 : loss = 0.4217385947704315
0.9375
epoch 40 : loss = 0.4257148802280426
0.90625
epoch 41 : loss = 0.2704349756240845
0.8984375
epoch 42 : loss = 0.1128798872232437
0.96875
epoch 43 : loss = 0.70346379728009053
0.9375
epoch 44 : loss = 0.7749876976013184
0.9375
epoch 45 : loss = 0.17467522621154785
0.8515625
epoch 46 : loss = 0.34885668754577637
0.9140625
epoch 47 : loss = 0.2769843339920044
0.9296875
epoch 48 : loss = 0.08638912439346313
0.9296875
epoch 49 : loss = 0.02232661098241806
0.9140625
epoch 50 : loss = 0.24123872816562653
0.9140625
Training is finished. Training time = 5.5392533302307125 minutes.

Process finished with exit code 0
```

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)

```

epoch 32 : Loss: 0.2913621219411513 Accuracy: 0.89625
Epoch 33 : Loss: 0.304840596832350976 Accuracy: 0.8984375
Epoch 34 : Loss: 0.2983116417870683 Accuracy: 0.8671875
Epoch 35 : Loss: 0.2861460124663377 Accuracy: 0.8984375
Epoch 36 : Loss: 0.29584261954408564 Accuracy: 0.9140625
Epoch 37 : Loss: 0.303260685022232 Accuracy: 0.90625
Epoch 38 : Loss: 0.2359452618554487 Accuracy: 0.8984375
Epoch 39 : Loss: 0.2525715239472308 Accuracy: 0.8828125
Epoch 40 : Loss: 0.2541556997319399 Accuracy: 0.8828125
Epoch 41 : Loss: 0.27864633072685389 Accuracy: 0.9140625
Epoch 42 : Loss: 0.2567639000461263 Accuracy: 0.921875
Epoch 43 : Loss: 0.26481134883301743 Accuracy: 0.921875
Epoch 44 : Loss: 0.2438070515833668 Accuracy: 0.9453125
Epoch 45 : Loss: 0.25241168935672714 Accuracy: 0.8671875
Epoch 46 : Loss: 0.29766080714865376 Accuracy: 0.9140625
Epoch 47 : Loss: 0.2540528971891282 Accuracy: 0.9140625
Epoch 48 : Loss: 0.2388009591443558 Accuracy: 0.890625
Epoch 49 : Loss: 0.2638074972960042 Accuracy: 0.9140625
Epoch 50 : Loss: 0.25470342278733094 Accuracy: 0.9140625
Epoch 51 : Loss: 0.2213110877674515 Accuracy: 0.921875
Epoch 52 : Loss: 0.19079741873478484 Accuracy: 0.9453125
Epoch 53 : Loss: 0.20032151592737538 Accuracy: 0.90625
Epoch 54 : Loss: 0.2286954752595748 Accuracy: 0.921875
Epoch 55 : Loss: 0.22085475091333107 Accuracy: 0.9375
Epoch 56 : Loss: 0.21728673378416039 Accuracy: 0.9296875
Epoch 57 : Loss: 0.20570847938470112 Accuracy: 0.90625
Epoch 58 : Loss: 0.2272556292868662 Accuracy: 0.890625
Epoch 59 : Loss: 0.2506937369767387 Accuracy: 0.89625
Training is finished. Training time = 19.891593194007875 minutes.

Process finished with exit code 0
```

```

epoch 47 : loss = 0.8950908236885971
0.9140625
epoch 48 : loss = 0.5763358473777771
0.8359375
epoch 49 : loss = 0.19641990959644318
0.84375
epoch 50 : loss = 0.3308316171169281
0.828125
epoch 51 : loss = 0.8173934817314148
0.84375
epoch 52 : loss = 0.24066342413425446
0.8359375
epoch 53 : loss = 0.4642934799194336
0.921875
epoch 54 : loss = 0.5331406593322754
0.8671875
epoch 55 : loss = 0.11389767378568649
0.875
epoch 56 : loss = 0.4446692168712616
0.875
epoch 57 : loss = 0.6288116574287415
0.8671875
epoch 58 : loss = 0.9575092792510986
0.8671875
epoch 59 : loss = 0.18718813359737396
0.84375
epoch 60 : loss = 0.4222375154495239
0.859375
Training is finished. Training time = 34.36792547305425 minutes.

Process finished with exit code 0
```

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).

```

↑ Epoch 22 Loss: 2.5585891151641585 Accuracy: 0.1484375
↓ Epoch 23 Loss: 2.558527198884877 Accuracy: 0.1484375
↔ Epoch 24 Loss: 2.556578199742204 Accuracy: 0.1484375
↔ Epoch 25 Loss: 2.556317935379805 Accuracy: 0.1484375
↔ Epoch 26 Loss: 2.5549014422853116 Accuracy: 0.1484375
↔ Epoch 27 Loss: 2.554621868214365 Accuracy: 0.1484375
↔ Epoch 28 Loss: 2.5557435708225 Accuracy: 0.1484375
↔ Epoch 29 Loss: 2.555232383437076 Accuracy: 0.1484375
↔ Epoch 30 Loss: 2.555475467275485 Accuracy: 0.1484375
↔ Epoch 31 Loss: 2.55573072473816 Accuracy: 0.1484375
↔ Epoch 32 Loss: 2.5551345397948226 Accuracy: 0.1484375
↔ Epoch 33 Loss: 2.5585416757454307 Accuracy: 0.1484375
↔ Epoch 34 Loss: 2.557353605658321 Accuracy: 0.1484375
↔ Epoch 35 Loss: 2.5549917019019692 Accuracy: 0.1484375
↔ Epoch 36 Loss: 2.5549356351464483 Accuracy: 0.1484375
↔ Epoch 37 Loss: 2.556123553671786 Accuracy: 0.1484375
↔ Epoch 38 Loss: 2.555984133845184 Accuracy: 0.1484375
↔ Epoch 39 Loss: 2.5561842999215854 Accuracy: 0.1484375
↔ Epoch 40 Loss: 2.557039315417663 Accuracy: 0.1484375
↔ Epoch 41 Loss: 2.5535852929293097 Accuracy: 0.1484375
↔ Epoch 42 Loss: 2.555161963074894 Accuracy: 0.1484375
↔ Epoch 43 Loss: 2.5544788009029324 Accuracy: 0.1484375
↔ Epoch 44 Loss: 2.55203213934171 Accuracy: 0.1484375
↔ Epoch 45 Loss: 2.5536156064372952 Accuracy: 0.1484375
↔ Epoch 46 Loss: 2.5536655854370633 Accuracy: 0.1484375
↔ Epoch 47 Loss: 2.55309506892405 Accuracy: 0.1484375
↔ Epoch 48 Loss: 2.554008689977359 Accuracy: 0.1484375
↔ Epoch 49 Loss: 2.5552275019176935 Accuracy: 0.1484375
Training is finished. Training time = 6.205594571431478 minutes.

Process finished with exit code 0
```



```

↑ epoch 37 : loss = 2.708050012588501
↓ epoch 38 : loss = 2.708050012588501
↔ epoch 39 : loss = 2.708050012588501
↔ epoch 40 : loss = 2.708050012588501
↔ epoch 41 : loss = 2.708050012588501
↔ epoch 42 : loss = 2.708050012588501
↔ epoch 43 : loss = 2.708050012588501
↔ epoch 44 : loss = 2.708050012588501
↔ epoch 45 : loss = 2.708050012588501
↔ epoch 46 : loss = 2.708050012588501
↔ epoch 47 : loss = 2.708050012588501
↔ epoch 48 : loss = 2.708050012588501
↔ epoch 49 : loss = 2.708050012588501
↔ epoch 50 : loss = 2.708050012588501
Training is finished. Training time = 5.883743647734324 minutes.

Process finished with exit code 0
```

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)

```

↑ Epoch 22 Loss: 2.708050012588501 Accuracy: 0.1015625
↓ Epoch 23 Loss: 2.708050012588501 Accuracy: 0.1015625
Epoch 24 Loss: 2.708050012588501 Accuracy: 0.1015625
Epoch 25 Loss: 2.708050012588501 Accuracy: 0.1015625
Epoch 26 Loss: 2.708050012588501 Accuracy: 0.1015625
Epoch 27 Loss: 2.708050012588501 Accuracy: 0.1015625
Epoch 28 Loss: 2.708050012588501 Accuracy: 0.1015625
Epoch 29 Loss: 2.708050012588501 Accuracy: 0.1015625
Epoch 30 Loss: 2.708050012588501 Accuracy: 0.1015625
Epoch 31 Loss: 2.708050012588501 Accuracy: 0.1015625
Epoch 32 Loss: 2.708050012588501 Accuracy: 0.1015625
Epoch 33 Loss: 2.708050012588501 Accuracy: 0.1015625
Epoch 34 Loss: 2.708050012588501 Accuracy: 0.1015625
Epoch 35 Loss: 2.708050012588501 Accuracy: 0.1015625
Epoch 36 Loss: 2.708050012588501 Accuracy: 0.1015625
Epoch 37 Loss: 2.708050012588501 Accuracy: 0.1015625
Epoch 38 Loss: 2.708050012588501 Accuracy: 0.1015625
Epoch 39 Loss: 2.708050012588501 Accuracy: 0.1015625
Epoch 40 Loss: 2.708050012588501 Accuracy: 0.1015625
Epoch 41 Loss: 2.708050012588501 Accuracy: 0.1015625
Epoch 42 Loss: 2.708050012588501 Accuracy: 0.1015625
Epoch 43 Loss: 2.708050012588501 Accuracy: 0.1015625
Epoch 44 Loss: 2.708050012588501 Accuracy: 0.1015625
Epoch 45 Loss: 2.708050012588501 Accuracy: 0.1015625
Epoch 46 Loss: 2.708050012588501 Accuracy: 0.1015625
Epoch 47 Loss: 2.708050012588501 Accuracy: 0.1015625
Epoch 48 Loss: 2.708050012588501 Accuracy: 0.1015625
Epoch 49 Loss: 2.708050012588501 Accuracy: 0.1015625
Training is finished. Training time = 6.899254659811656 minutes.

Process finished with exit code 0
```



```

↑ epoch 37 : loss = 0.0546875
↓ epoch 38 : loss = 2.708050012588501
Epoch 39 : loss = 2.708050012588501
epoch 40 : loss = 2.708050012588501
0.0546875
epoch 41 : loss = 2.708050012588501
0.0546875
epoch 42 : loss = 2.708050012588501
0.0546875
epoch 43 : loss = 2.708050012588501
0.0546875
epoch 44 : loss = 2.708050012588501
0.0546875
epoch 45 : loss = 2.708050012588501
0.0546875
epoch 46 : loss = 2.708050012588501
0.0546875
epoch 47 : loss = 2.708050012588501
0.0546875
epoch 48 : loss = 2.708050012588501
0.0546875
epoch 49 : loss = 2.708050012588501
0.0546875
epoch 50 : loss = 2.708050012588501
0.0546875
Training is finished. Training time = 6.806599708398183 minutes.

Process finished with exit code 0
```

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).

```

↑ Epoch 22 Loss: 0.33401143089947855 Accuracy: 0.875
↓ Epoch 23 Loss: 0.37453232137328485 Accuracy: 0.8671875
Epoch 24 Loss: 0.3159793592610402 Accuracy: 0.890425
Epoch 25 Loss: 0.2762102754310566 Accuracy: 0.8828125
Epoch 26 Loss: 0.2812619451749122 Accuracy: 0.914625
Epoch 27 Loss: 0.24653672161748855 Accuracy: 0.921875
Epoch 28 Loss: 0.2388854310295339 Accuracy: 0.921875
Epoch 29 Loss: 0.2535094980592445 Accuracy: 0.8984375
Epoch 30 Loss: 0.230536197733267 Accuracy: 0.9453125
Epoch 31 Loss: 0.25285540154929886 Accuracy: 0.914625
Epoch 32 Loss: 0.223354766389942 Accuracy: 0.9296875
Epoch 33 Loss: 0.20861178820392835 Accuracy: 0.921875
Epoch 34 Loss: 0.18749857798314983 Accuracy: 0.953125
Epoch 35 Loss: 0.193278783224514 Accuracy: 0.953125
Epoch 36 Loss: 0.19020764911705154 Accuracy: 0.9375
Epoch 37 Loss: 0.18197085575784666 Accuracy: 0.9453125
Epoch 38 Loss: 0.188654670360832 Accuracy: 0.9375
Epoch 39 Loss: 0.1725307997434033 Accuracy: 0.9453125
Epoch 40 Loss: 0.168306197934363 Accuracy: 0.953125
Epoch 41 Loss: 0.1810482321149212 Accuracy: 0.914625
Epoch 42 Loss: 0.1911467423681627 Accuracy: 0.914625
Epoch 43 Loss: 0.217566143089369 Accuracy: 0.914625
Epoch 44 Loss: 0.2597896139601848 Accuracy: 0.890625
Epoch 45 Loss: 0.22721647022891853 Accuracy: 0.9296875
Epoch 46 Loss: 0.209372430611322 Accuracy: 0.921875
Epoch 47 Loss: 0.18607152758513468 Accuracy: 0.96875
Epoch 48 Loss: 0.16981095293442072 Accuracy: 0.921875
Epoch 49 Loss: 0.17715916842601057 Accuracy: 0.955125
Training is finished. Training time = 8.53837239742279 minutes.

Process finished with exit code 0
```



```

↑ epoch 37 : loss = 0.9296875
↓ epoch 38 : loss = 0.5514474511146545
0.8984375
epoch 39 : loss = 0.12219281697273254
0.9296875
epoch 40 : loss = 0.4239218235015869
0.9375
epoch 41 : loss = 0.16853214800357817
0.896625
epoch 42 : loss = 0.015416372567415237
0.9296875
epoch 43 : loss = 0.30792751908302307
0.9375
epoch 44 : loss = 0.5682316422462463
0.914625
epoch 45 : loss = 0.07416034489870071
0.9375
epoch 46 : loss = 0.1434275358915329
0.90625
epoch 47 : loss = 0.1574176400899887
0.9689375
epoch 48 : loss = 0.07072959840297699
0.9453125
epoch 49 : loss = 0.13403262197971344
0.90625
epoch 50 : loss = 0.5787031650543213
0.953125
Training is finished. Training time = 8.727361881732941 minutes.

Process finished with exit code 0
```

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)

```

Epoch 22 Loss: 1.1512936737577795 Accuracy: 0.6796875
Epoch 23 Loss: 1.2077052280054255 Accuracy: 0.578125
Epoch 24 Loss: 1.1748638021743905 Accuracy: 0.671875
Epoch 25 Loss: 1.3440484429828011 Accuracy: 0.578125
Epoch 26 Loss: 1.357517209934495 Accuracy: 0.578125
Epoch 27 Loss: 1.29201416292433 Accuracy: 0.6171875
Epoch 28 Loss: 1.2464502686161105 Accuracy: 0.625
Epoch 29 Loss: 1.32335388717961908 Accuracy: 0.5234375
Epoch 30 Loss: 1.556827971983375 Accuracy: 0.693975
Epoch 31 Loss: 1.3831628975221666 Accuracy: 0.5859375
Epoch 32 Loss: 1.3944447581040658 Accuracy: 0.53125
Epoch 33 Loss: 1.3376211054244285 Accuracy: 0.6484375
Epoch 34 Loss: 1.3201656164759297 Accuracy: 0.7109375
Epoch 35 Loss: 1.2236057398682934 Accuracy: 0.546875
Epoch 36 Loss: 1.396875830199347 Accuracy: 0.5859375
Epoch 37 Loss: 1.3118163842470853 Accuracy: 0.425
Epoch 38 Loss: 1.3324148582170726 Accuracy: 0.59375
Epoch 39 Loss: 1.3343086151753443 Accuracy: 0.5625
Epoch 40 Loss: 1.3466555824724293 Accuracy: 0.546875
Epoch 41 Loss: 1.3612992207882768 Accuracy: 0.4171875
Epoch 42 Loss: 1.3591521639945143 Accuracy: 0.53125
Epoch 43 Loss: 1.2723678981853743 Accuracy: 0.699375
Epoch 44 Loss: 1.2996882949395907 Accuracy: 0.578125
Epoch 45 Loss: 1.3087741894237066 Accuracy: 0.6484375
Epoch 46 Loss: 1.296152277518127 Accuracy: 0.5859375
Epoch 47 Loss: 1.3824680977958743 Accuracy: 0.6640625
Epoch 48 Loss: 1.3706727306640084 Accuracy: 0.59375
Epoch 49 Loss: 1.47894649034729 Accuracy: 0.59375
Training is finished. Training time = 6.19782178401947 minutes.

Process finished with exit code 0

```

```

epoch 37 : loss = 1.394545555114746
0.7109375
epoch 38 : loss = 1.3343467712402344
0.7734375
epoch 39 : loss = 0.958417452235413
0.6875
epoch 40 : loss = 0.763598235594635
0.6640625
epoch 41 : loss = 1.1018961668014526
0.71875
epoch 42 : loss = 1.2311229705810547
0.6875
epoch 43 : loss = 0.7908194661140442
0.6953125
epoch 44 : loss = 0.7185080647468567
0.65625
epoch 45 : loss = 0.8803557753562927
0.671875
epoch 46 : loss = 1.0141226053237915
0.6484375
epoch 47 : loss = 1.81777004272461
0.59375
epoch 48 : loss = 2.0399162769317627
0.5
epoch 49 : loss = 1.506697177886963
0.453125
epoch 50 : loss = 1.685970664024353
0.5
Training is finished. Training time = 6.170154615243276 minutes.

Process finished with exit code 0

```

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).

```

↑ Epoch 22 Loss: 2.708050012588501 Accuracy: 0.0859375
↓ Epoch 23 Loss: 2.708050012588501 Accuracy: 0.0859375
Epoch 24 Loss: 2.708050012588501 Accuracy: 0.0859375
Epoch 25 Loss: 2.708050012588501 Accuracy: 0.0859375
Epoch 26 Loss: 2.708050012588501 Accuracy: 0.0859375
Epoch 27 Loss: 2.708050012588501 Accuracy: 0.0859375
Epoch 28 Loss: 2.708050012588501 Accuracy: 0.0859375
Epoch 29 Loss: 2.708050012588501 Accuracy: 0.0859375
Epoch 30 Loss: 2.708050012588501 Accuracy: 0.0859375
Epoch 31 Loss: 2.708050012588501 Accuracy: 0.0859375
Epoch 32 Loss: 2.708050012588501 Accuracy: 0.0859375
Epoch 33 Loss: 2.708050012588501 Accuracy: 0.0859375
Epoch 34 Loss: 2.708050012588501 Accuracy: 0.0859375
Epoch 35 Loss: 2.708050012588501 Accuracy: 0.0859375
Epoch 36 Loss: 2.708050012588501 Accuracy: 0.0859375
Epoch 37 Loss: 2.708050012588501 Accuracy: 0.0859375
Epoch 38 Loss: 2.708050012588501 Accuracy: 0.0859375
Epoch 39 Loss: 2.708050012588501 Accuracy: 0.0859375
Epoch 40 Loss: 2.708050012588501 Accuracy: 0.0859375
Epoch 41 Loss: 2.708050012588501 Accuracy: 0.0859375
Epoch 42 Loss: 2.708050012588501 Accuracy: 0.0859375
Epoch 43 Loss: 2.708050012588501 Accuracy: 0.0859375
Epoch 44 Loss: 2.708050012588501 Accuracy: 0.0859375
Epoch 45 Loss: 2.708050012588501 Accuracy: 0.0859375
Epoch 46 Loss: 2.708050012588501 Accuracy: 0.0859375
Epoch 47 Loss: 2.708050012588501 Accuracy: 0.0859375
Epoch 48 Loss: 2.708050012588501 Accuracy: 0.0859375
Epoch 49 Loss: 2.708050012588501 Accuracy: 0.0859375
Training is finished. Training time = 6.99440973971161 minutes.

Process finished with exit code 0
```



```

↑ epoch 37 : loss = 2.708050012588501
0.0234375
↓ epoch 38 : loss = 2.708050012588501
0.0234375
epoch 39 : loss = 2.708050012588501
0.0234375
epoch 40 : loss = 2.708050012588501
0.0234375
epoch 41 : loss = 2.708050012588501
0.0234375
epoch 42 : loss = 2.708050012588501
0.0234375
epoch 43 : loss = 2.708050012588501
0.0234375
epoch 44 : loss = 2.708050012588501
0.0234375
epoch 45 : loss = 2.708050012588501
0.0234375
epoch 46 : loss = 2.708050012588501
0.0234375
epoch 47 : loss = 2.708050012588501
0.0234375
epoch 48 : loss = 2.708050012588501
0.0234375
epoch 49 : loss = 2.708050012588501
0.0234375
epoch 50 : loss = 2.708050012588501
0.0234375
Training is finished. Training time = 6.796968098481496 minutes.

Process finished with exit code 0
```

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.

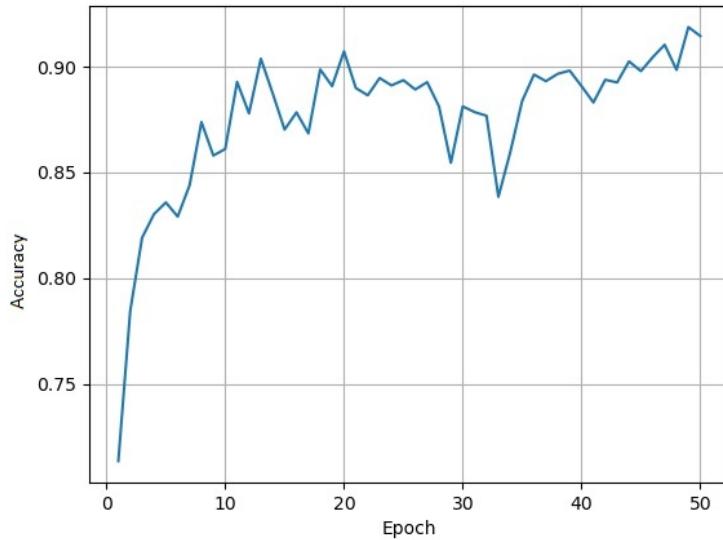


Figure 16: Accuracy of the trained network in each epoch

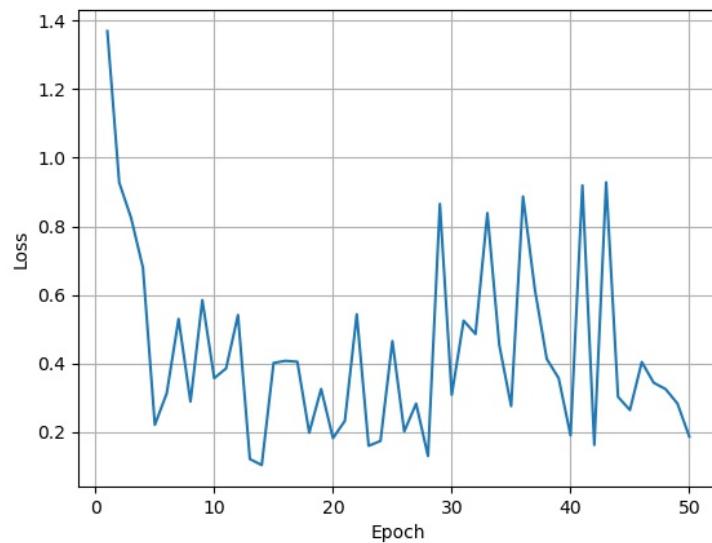


Figure 17: Loss of the trained network in each epoch

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:

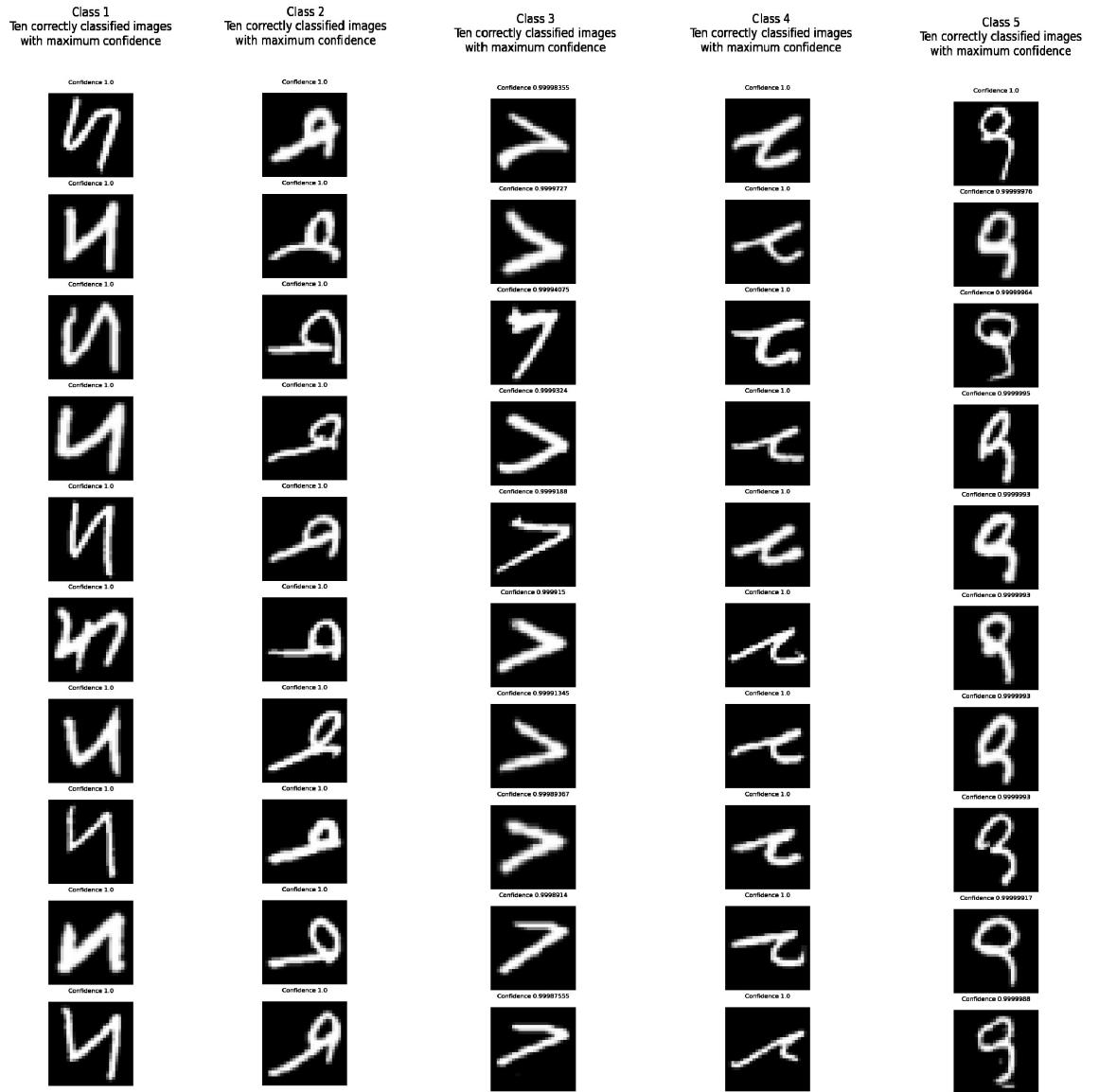


Figure 19: Ten correctly classified images with maximum confidence for 1-5 classes

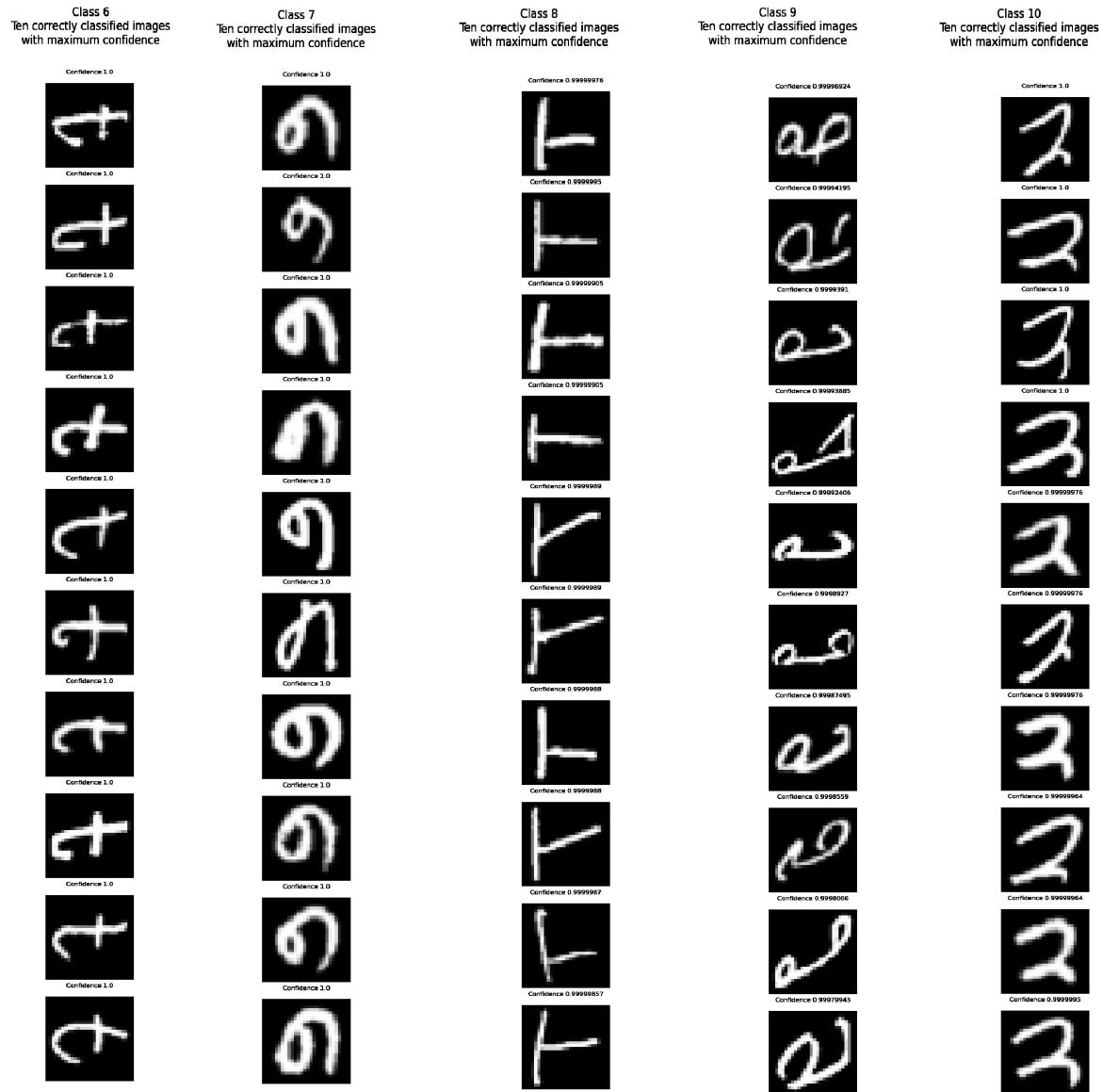


Figure 20: Ten correctly classified images with maximum confidence for 6-10 classes

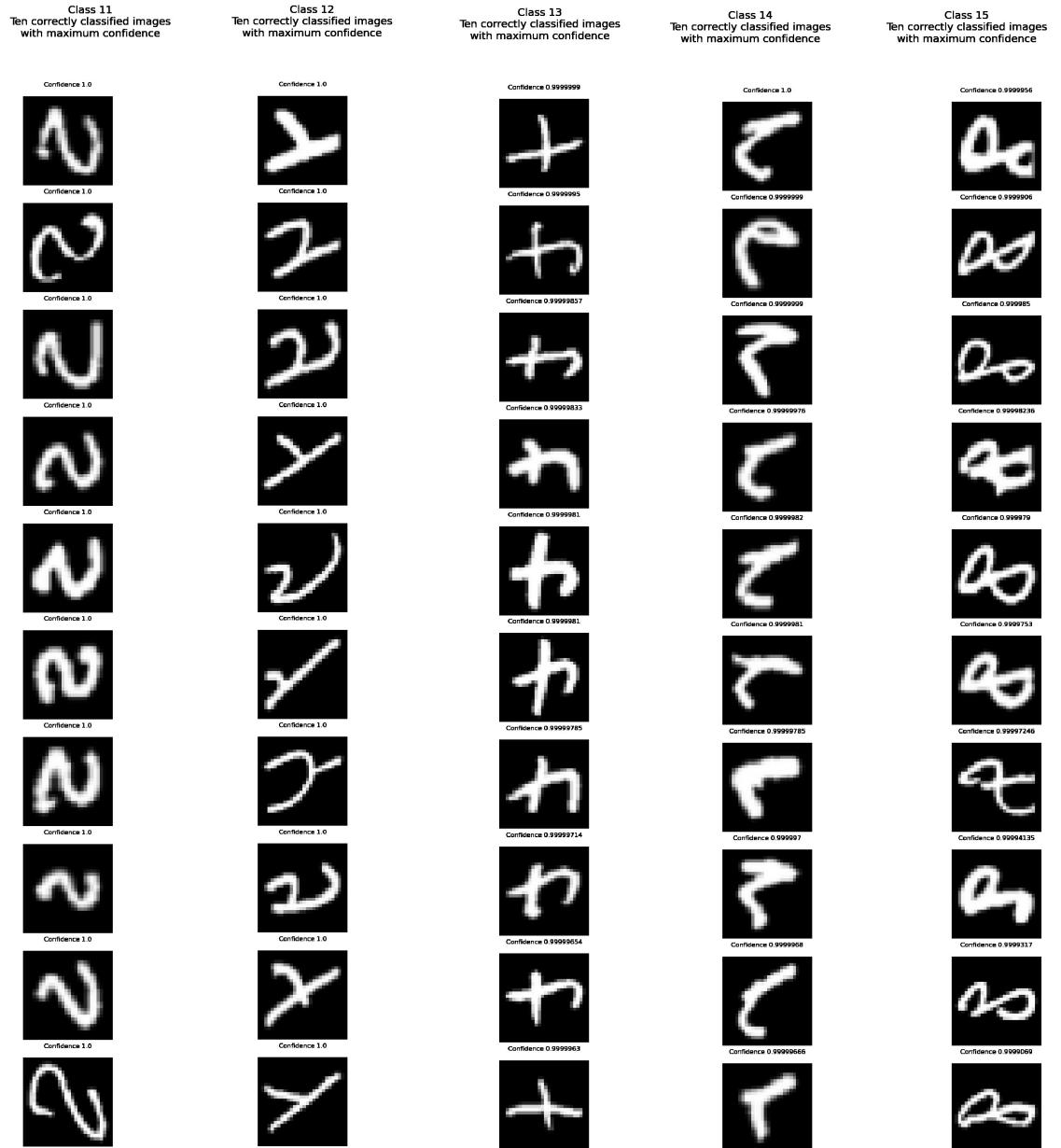


Figure 21: Ten correctly classified images with maximum confidence for 11-15 classes

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

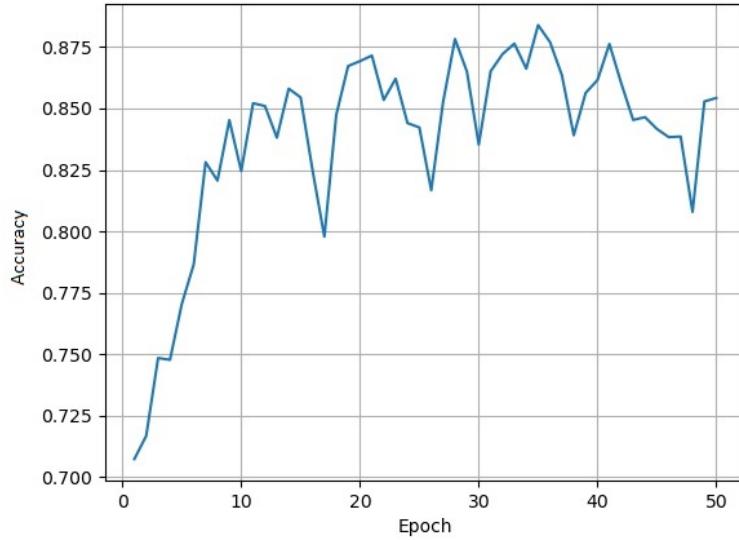


Figure 23: Accuracy of the trained network in each epoch in our model with 2D Batch Normalization

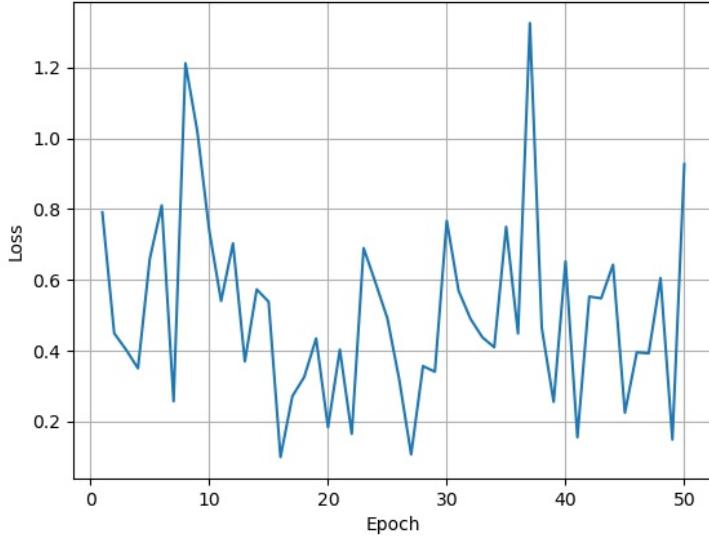


Figure 24: Loss of the trained network in each epoch in our model with 2D Batch Normalization

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.