Computer Vision
Assignment# 1

Submitted by
Shubham Kumar(21-944-491)

22 OCTOBER, 2021

Contents

1 Simple 2D classifier

1.1 *2.3 Training loop*

Accuracy achieved with Linear classifier was 47.6% , which is expected because ind 2-d plane we cannot find a straight line which can separate the regions (characterized by the two classes). The best that can be done is to either make the line pass through the center or let the line be outside of the both circular regions (in either case accuracy would be poor and close to random guess , assuming the number of data-points for both the classes are almost same).

1.2 *2.4 Multi-layer perceptron*

With Multi-layer perceptron, the accuracy achieved was 99.8% . It was possible because of the addition of hidden layers and the non-linear activation function. With addition of these two to the model, the model was now able to internally transform the input features to higher dimensional feature space where it is easy to find a set of parameters which can model the distribution of the data and thus find a separating boundary. Hence, the result was much better than the linear classifier.

1.3 *2.5 Feature transform*

I chose polar coordinate system (r, θ) based on the given data. Since, the two classes are on concentric circles with different radii and center as origin, therefore, in polar coordinates the data points for these two classes will have significantly different values of r for any given θ. Thus in (r, θ) space a simple linear classifier can also model the line separating the two classes. With this transformed feature and linear classifier, the accuracy achieved was: 92.4%

2 Digit classifier

2.1 *3.3 Multi-layer perceptron*

The following are the accuracies obtained for:
when using single layer: 90.52%, and
when using "MLP with one hidden layer of dimension 32 followed byReLU and then the final linear prediction layer" : 94.37%

2.2 *3.4 Convolutional network*

With this architecture (CNN based) , the accuracy obtained was: 98.14%

2.3 *3.5 Comparison of number of parameters*

For MLP model parameters:
Since the image is converted to a vector (of length $28{\times}28 = 784$) and hidden layer dimension is 32, therefore, the dimension of the first weight matrix will be 784×32, and the number of bias parameters will be 32

For the second (which is also the last) layer:
Number of weight parameters : 32×10 (32 is the output dimension of the previous layer and 10 is the output dimension of current layer) ,and Bias parameters : 10

So total parameters $= (784 \times 32 + 32) + (32 \times 10 + 10) = 25450$

```
Output of : [(n, p.numel()) for n, p in net.named_parameters() if p.requires_grad]
```

```
[('layers.0.weight', 25088), ('layers.0.bias', 32),
('layers.2.weight', 320), ('layers.2.bias', 10)]
```

For CNN Model parameters, we have:
First CNN layer: $1{\times}3{\times}3{\times}8$ (*i.e.* number of input channels $\times$ kernel dimension 1 $\times$ kernel dimension 2 $\times$ number of output channels weight parameters and 8 (number of output channels) bias parameters . Total of 80 parameters
Similarly,
Second CNN Layer: $8{\times}3{\times}3{\times}6 + 16$
Third CNN Layer : $16{\times}3{\times}3{\times}32 + 32$

For final Linear Layer: $32{\times}10 + 10$

So CNN based model has a total of 6218 parameters. (Note: ReLu and MaxPooling has no trainable parameters)

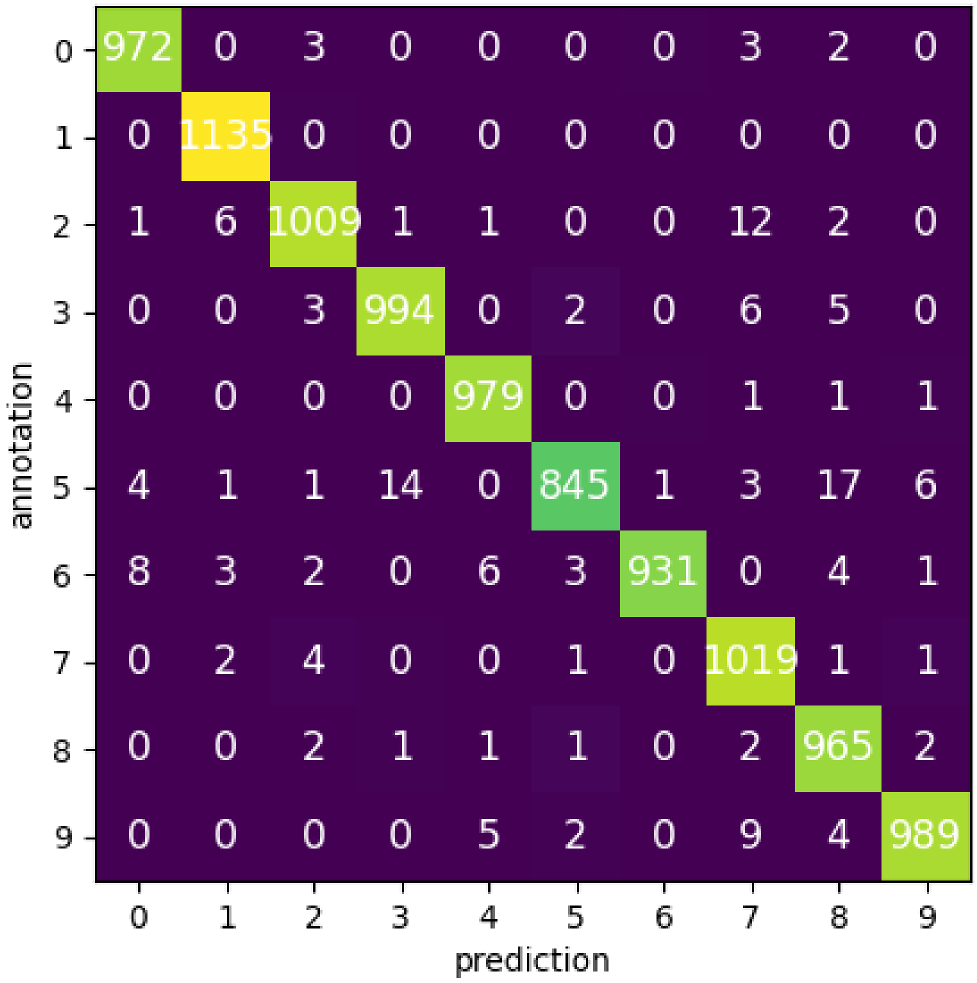
```
Output of : [(n, p.numel()) for n, p in net.named_parameters() if p.requires_grad]
```

```
[('layers.0.weight', 72), ('layers.0.bias', 8),
('layers.3.weight', 1152), ('layers.3.bias', 16),
('layers.6.weight', 4608), ('layers.6.bias', 32),
('classifier.0.weight', 320), ('classifier.0.bias', 10)]
```

As compared to MLP the CNN model has less number of parameters (about one-fourth of the number of MLP parameters), still CNN achieves better accuracy. CNN based model is able to learn spatial features (of the image) easily and with less number of parameters.

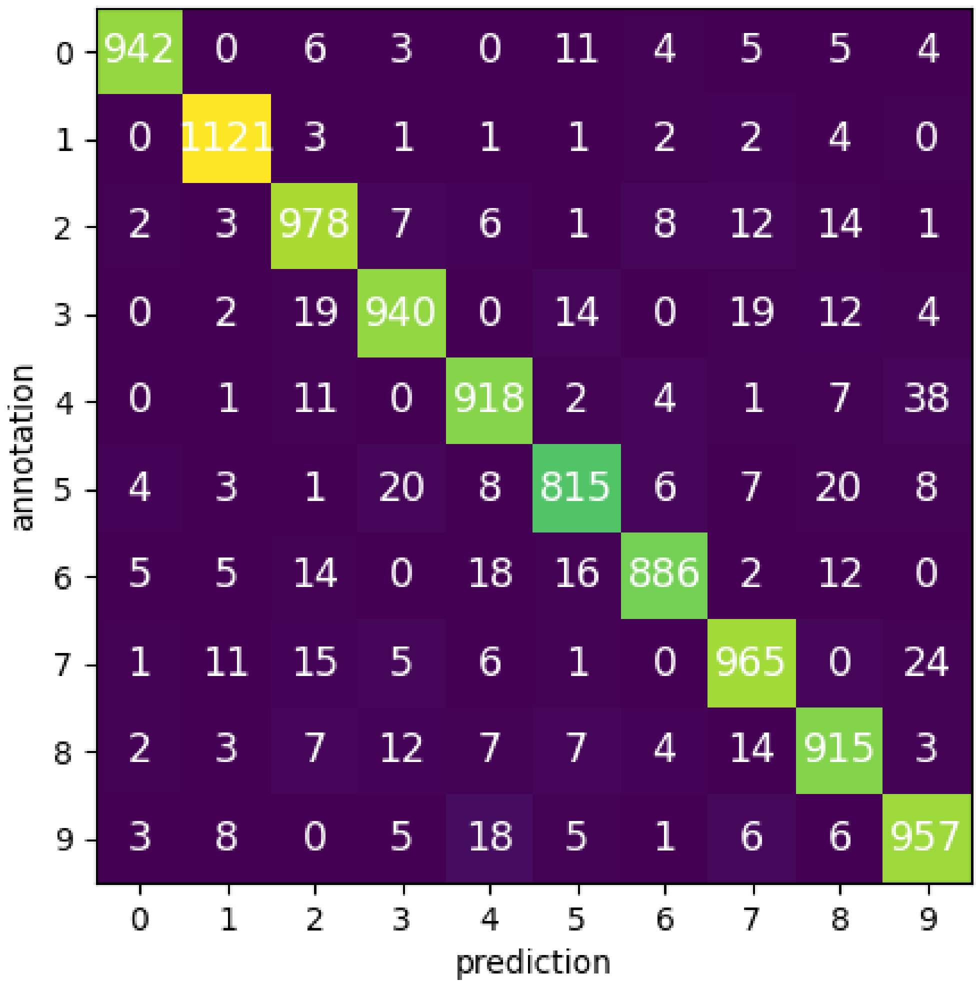
2.4 3.6 Confusion matrix

Figure 1 shows the Confusion Matrix for CNN Model. Most of the predictions are correct (as reflected by bright diagonal).
We observe that model confuses the number 5 the most. It is expected too because written instances of 3 and 8 may match significantly with 5 due to the similarity in shapes. Similar situation can be observed with 2 and 7, wherein due to resemblance in shape some occurrences of 2 may have been predicted as 7.



annotation \ prediction	0	1	2	3	4	5	6	7	8	9
0	972	0	3	0	0	0	0	3	2	0
1	0	1135	0	0	0	0	0	0	0	0
2	1	6	1009	1	1	0	0	12	2	0
3	0	0	3	994	0	2	0	6	5	0
4	0	0	0	0	979	0	0	1	1	1
5	4	1	1	14	0	845	1	3	17	6
6	8	3	2	0	6	3	931	0	4	1
7	0	2	4	0	0	1	0	1019	1	1
8	0	0	2	1	1	1	0	2	965	2
9	0	0	0	0	5	2	0	9	4	989

Figure 1: Confusion Matrix (for CNN Model)



annotation \ prediction	0	1	2	3	4	5	6	7	8	9
0	942	0	6	3	0	11	4	5	5	4
1	0	1121	3	1	1	1	2	2	4	0
2	2	3	978	7	6	1	8	12	14	1
3	0	2	19	940	0	14	0	19	12	4
4	0	1	11	0	918	2	4	1	7	38
5	4	3	1	20	8	815	6	7	20	8
6	5	5	14	0	18	16	886	2	12	0
7	1	11	15	5	6	1	0	965	0	24
8	2	3	7	12	7	7	4	14	915	3
9	3	8	0	5	18	5	1	6	6	957

Figure 2: Confusion Matrix (for MLP Model)