Part 1: Theory

Q1.1 - 5 Pts We have a function which takes a two-dimensional input x = (x1, x2) and has two parameters w = (w1, w2) given by f (x, w) = $\sigma(\sigma(x_1 + w_1)w_2 + x_2)$ where σ(x) = $\frac{1}{1+e^{-x}}$. We want to estimate the parameters that minimize a L-2 loss by performing gradient descent. We initialize both the parameters to 0. Assume that we are given a training point $x_1 = 1, x_2 = 0, y = 5$, where y is the true value at (x_1 ,x_2). Based on this answer the following questions:

(a) What is the value of $\frac{\partial f}{\partial w_2}$?

$$\frac{\partial f}{\partial w_2} = \frac{\partial f}{\partial((x_1+w_1)w_2+x_2)} \cdot \frac{\partial((x_1+w_1)w_2+x_2)}{\partial w_2}$$
$$= \sigma((x_1+w_1)w_2+x_2)\sigma(1-(x_1+w_1)w_2-x_2)\cdot(x_1+w_1)$$

plug in the values:

$\sigma((1+0)*0+0)\sigma(1-(1+0)*0-0)(1+0)$

$= \frac{1}{2(1+\frac{1}{e})}$

(b) If the learning rate is 0.5, what will be the value of w_2 after one update using SGD?

Predict Value:

$$\sigma(\sigma(x_1+w_1)\cdot 0+0) = \sigma(0) = \frac{1}{2}$$

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial f}\frac{\partial f}{\partial w_2}$$
$$= -\eta(y-\dot{y})\frac{\partial f}{\partial w_2}$$
$$= -\frac{1}{2}(5-0.5)\frac{1}{2(1+\frac{1}{e})}$$

$\therefore w_2 = w_2^{old} + -\frac{1}{2}(5-0.5)\frac{1}{2(1+\frac{1}{e})} = -\frac{1}{2}(5-0.5)\frac{1}{2(1+\frac{1}{e})}$

Q1.2 - 5 Pts All types of deep networks use non-linear activation functions for their hidden layers. Suppose we have a neural network (not a CNN) with input dimension N and output dimension C and T hidden layers. Prove that if we have a linear activation function g, then the number of hidden layers has no effect on the actual network.

Since g is a linear activation function, g is just a linear transformation between space. Note the hidden layers are just different forms of affine transformation, then

$Y = (g \circ f)^{(T)}(X)$ can just be write as $Y = AX + B$ where A is a $C \times N$ matrix for the transform of X and B is the bias at each level transform and summed up. Then no matter how many hidden layer is added the effect is just the same of having 1 layer.

Q1.3 - 5 Pts In training deep networks ReLU activation function is generally preferred to sigmoid, comment why?

ReLU is quicker then sigmoid, as ReLU is just max(0,x) which the estimation of sigmoid involve at least one division. The same applies for back propagation as well.

In deep network, sigmoid also have the problem of gradient vanishing. since $\sigma'(x) = \sigma(x)\sigma(1-x)$, the gradient is about 0 when $\sigma(x)$ is close to 0 or 1. Then when chain rule applies, the gradient tends to 0 when propagating backwards which leads to inefficient network. (ReLU may face the same problem for having dead nodes but less significantly then sigmoid, and can be solved using leaky ReLU or ELU)

Q1.4 - 5 Pts Why is it not a good idea to initialize a network with all zeros. How about all ones, or some other constant value?

The network should not start at all constant value(in most cases).

- The gradient of weights sometime depends on other weights. For example, If the first layer output all 0, then the gradient of w of second level $\frac{\partial L}{\partial w_2} = 0$ and does not change. This lead to slow and inefficient learning. (not sure but it seems that the network can't learn at all if we add a ReLU after it, which blocks all gradient)
- SGD search for minima. It is more likely to stuck at local minima when all values are the same, then the network fail to find global maxima and is suboptimal.
- If the layer doesn't have dropout layer or max pooling layer, the gradient of each node is the same over the layer, then all nodes in the layer will remain the same with other, then the layer is quite likely to be useless (The whole network may act as a same neuron).
- Nodes die out sometime(for example node the output negative value before reaching a ReLU layer). Nodes with similar value may die together, then the training is ineffective.
- In some cases, such as right before a ReLU layer, setting the bias to a constant 1 may help the network to start learning, as gradient can only pass when the otput of previous layer is positive.

Q1.5 - 5 Pts There are a lot of standard Convolutional Neural Network architectures used in the literature. In this question we will analyse the complexity of these networks measured in terms of the number of parameters. For each of the following networks calculate the total number of parameters.

- (a) AlexNet

$$\begin{gathered}
L_1 : 96 \times 11^2 \times 3 + 2 \times 48 = 34944 \\
(MaxPool) \\
L_2 : 2 \times 48 \times 5^2 \times 2 \times 128 + 2 \times 128 = 614656 \\
(MaxPool) \\
L_3 : 2 \times 128 \times 3^2 \times 2 \times 192 + 2 \times 192 = 885120 \\
L_4 : 2 \times 192 \times 3^2 \times 2 \times 192 + 2 \times 192 = 1327488 \\
L_5 : 2 \times 192 \times 3^2 \times 2 \times 128 + 2 \times 128 = 884992 \\
(MaxPool) \\
L_6 : 6 \times 6 \times 256 \times 4096 + 4096 = 37752832 \\
L_7 : 4096 \times 4096 + 4096 = 16781312 \\
L_8 : 4096 \times 1000 + 1000 = 4097000 \\
Total = 62378344 \sim 62M
\end{gathered}$$

- (b) VGG-16

  (given in paper)(in millions)

NetWork	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

- (c) GoogLeNet

(From Table 1) (2.7+112+159+380+364+437+463+580+840+1072+1388+1000)K

$\sim 6.8M$

Inception layers are used in GoogLENet, which concat convolution of different size layers (e.g. $5 \times 5, 3 \times 3, 1 \times 1$) to form the next Layer. At each level there are less number of kernel is smaller then other network(~64 5X5 kernel, 256 3x3 kernel, compared with AlexNet with 256 or 364 5X5 kernel per level, which multiple up between levels). Also, Average pooling is used to replace a fully connected layer where averaging pooling have no parameters.

Part 4: Training

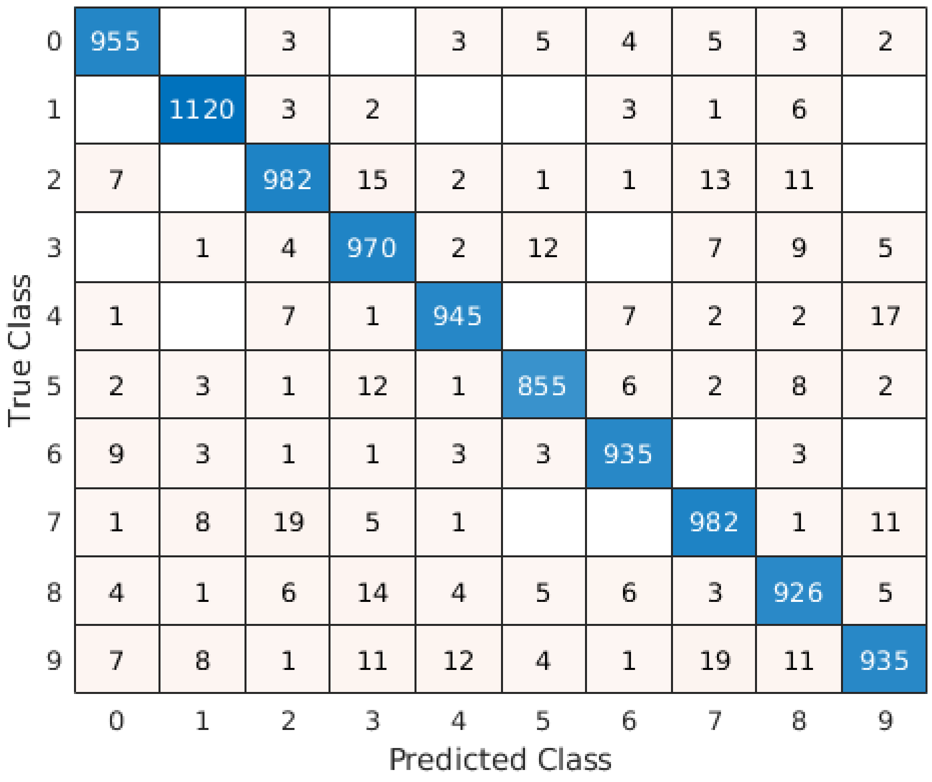
Q4.1 Training - 5 Pts

The script train lenet.m defines the optimization parameters and per-
forms the actual updates on the network. This script loads a pretrained network and trains
the network for 2000 iterations. Report the test accuracy obtained in your write-up after
training for 3000 more iterations.

test accuracy: 0.976

Q4.2 Test the network - 10 Pts

The script test lenet.m has been provided which runs the test data through the network and obtains the predictions probabilities. Modify this script to generate the confusion matrix and comment on the top two confused pairs of classes.



True Class \ Predicted Class	0	1	2	3	4	5	6	7	8	9
0	955		3		3	5	4	5	3	2
1		1120	3	2			3	1	6	
2	7		982	15	2	1	1	13	11	
3		1	4	970	2	12		7	9	5
4	1		7	1	945		7	2	2	17
5	2	3	1	12	1	855	6	2	8	2
6	9	3	1	1	3	3	935		3	
7	1	8	19	5	1			982	1	11
8	4	1	6	14	4	5	6	3	926	5
9	7	8	1	11	12	4	1	19	11	935

Most confused pair: 7 confused as 2, and 9 confused as 7

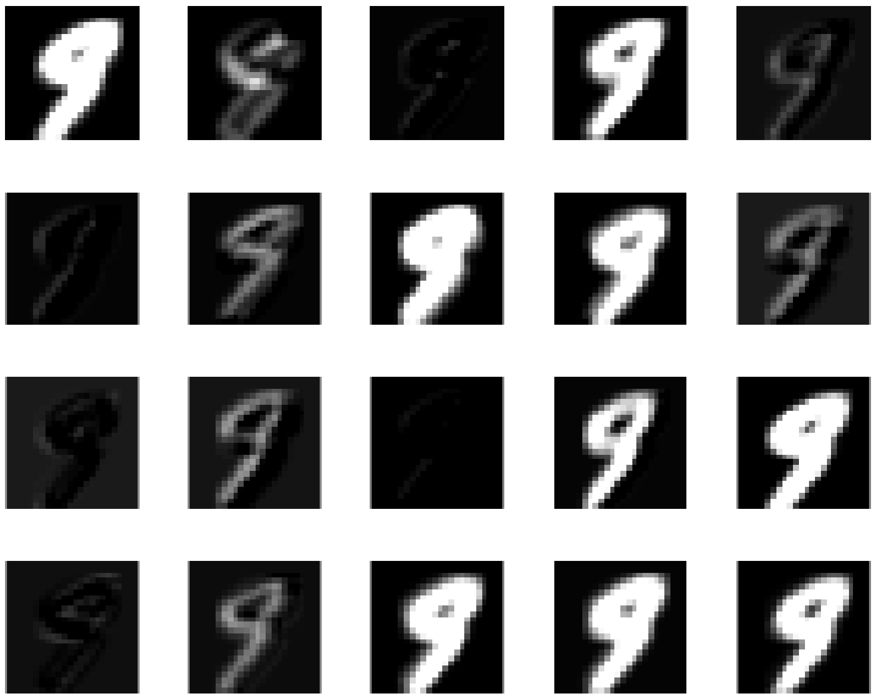
I guess this is acceptable as the shape of $2, 7, 9$ look similar in some written form, and some of the cases in the data set seems to be able to confuse human(me) as well.

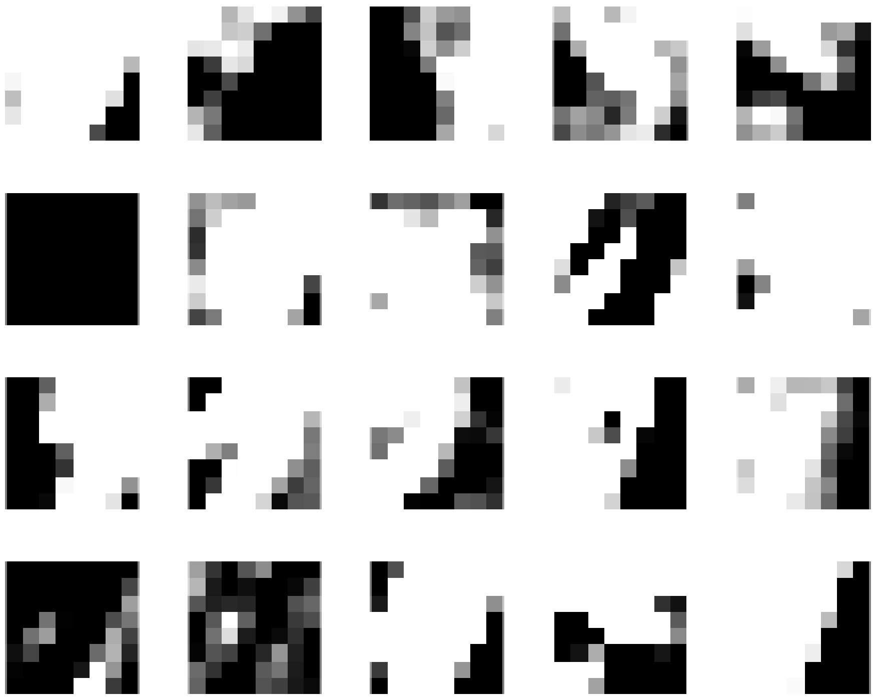
Part 5: Visualization

Q5.1 - 5 Pts

Write a script vis data.m which can load a sample image from the data, visualize the output of the second and third layers. Show 20 images from each layer on a single figure
file (use subplot and organize them in 4 × 5 format - like in Figure 4).

(I assume that the second and the third conv net is needed, not the ReLU or pooling layer)





Q5.2 - 5 Pts

Compare the feature maps to the original image and explain the differences.

The original shape can still be seen in the second layer, but it is hardly found in the third layer.

Seems that different kind of edges appear in layer 2, where some inverse of edges and boded edge also appear.

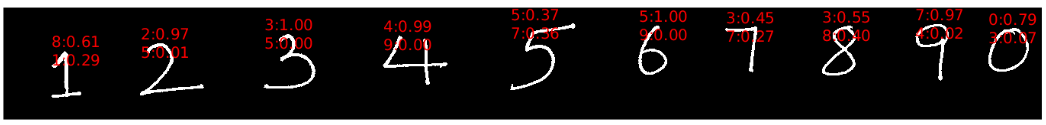
Some shapes appear in layer 3, such as circle and a vertical stroke.

Part 6: Image Classification - 20 pts

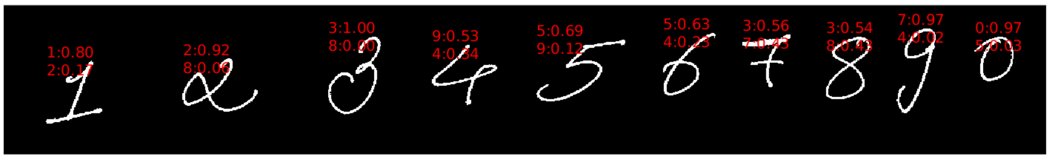
Results:

There might be errors in the recognition, report the output of your network in the report.

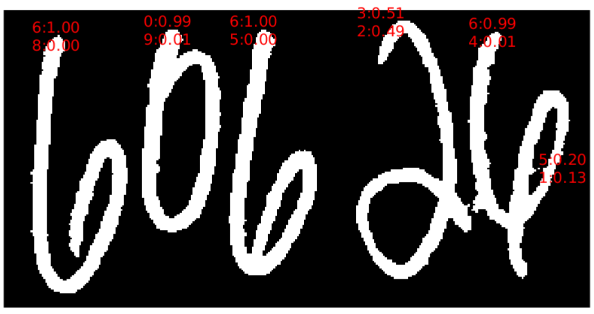
In the above images, the two best prediction are shown, with the output of soft max probability after them.



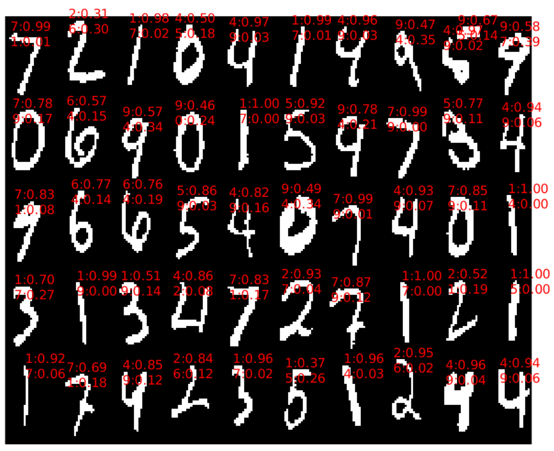
Digit 6 and digit 9 are very wrong. digit 7 is also wrong but at least 7 is its second prediction.



Digit 4, 6, 7, 8 and 9 are wrong. Digit 8 seems to be confused with 3 and Digit 9 seems to be confused as 7 quite often.



My algorithm make the last digit into 2. Even I can't tell is it 4 or 6 though..



7:0.99
1:0.01
2:0.31
6:0.30
1:0.98
7:0.02
4:0.50
5:0.18
4:0.97
9:0.03
1:0.99
7:0.01
4:0.96
9:0.03
9:0.47
4:0.35
9:0.67
4:0.97
5:0.14
9:0.02
9:0.58
7:0.39
7:0.78
9:0.17
6:0.57
4:0.15
9:0.57
4:0.34
9:0.46
0:0.24
1:1.00
7:0.00
5:0.92
9:0.03
9:0.78
4:0.21
7:0.99
9:0.00
5:0.77
9:0.11
4:0.94
9:0.06
7:0.83
1:0.08
6:0.77
4:0.14
6:0.76
4:0.19
5:0.86
9:0.03
4:0.82
9:0.16
9:0.49
4:0.34
7:0.99
9:0.01
4:0.93
9:0.07
7:0.85
9:0.11
1:1.00
4:0.00
1:0.70
7:0.27
1:0.99
9:0.00
1:0.51
9:0.14
4:0.86
2:0.08
7:0.83
1:0.17
2:0.93
7:0.04
7:0.87
9:0.12
1:1.00
7:0.00
2:0.52
1:0.19
1:1.00
5:0.00
1:0.92
7:0.06
7:0.69
1:0.18
4:0.85
9:0.12
2:0.84
6:0.12
1:0.96
7:0.02
1:0.37
5:0.26
1:0.96
4:0.03
2:0.95
6:0.02
4:0.96
9:0.04
4:0.94
9:0.06