# 1 Theory

## Q1.1

(a) Given

- $f(x, w) = \sigma(\sigma(x_1 w_1)w_2 + x_2)$
- $\sigma(x) = \dfrac{1}{1+e^{-x}}$ with $\sigma'(x) = \dfrac{-1(-e^{-x})}{(1+e^{-x})^2} = \dfrac{1}{1+e^{-x}}\dfrac{(1+e^{-x})-(1)}{1+e^{-x}} = \sigma(x)(1-\sigma(x))$

$$\begin{aligned}\frac{\partial f}{\partial w_2} &= \sigma(\sigma(x_1 w_1)w_2 + x_2)(1 - \sigma(\sigma(x_1 w_1)w_2 + x_2)) \cdot \sigma(x_1 w_1) \\ &= \sigma(\sigma(1 \cdot 0) \cdot 0 + 0)(1 - \sigma(\sigma(1 \cdot 0) \cdot 0 + 0)) \cdot \sigma(1 \cdot 0) \\ &= (0.5)(0.5) \cdot 0.5 \\ &= 0.125\end{aligned}$$

(b) A forward pass gives $\hat{y} = \sigma(\sigma(1 \cdot 0) \cdot 0 + 0) = 0.5$. L2 loss $\mathcal{L}(y, \hat{y}) = (y - \hat{y})^2$, hence $\dfrac{\partial \mathcal{L}}{\partial f} = 2(y - \hat{y}) \cdot (-1) = -2(5 - 0.5) = -9$.

$$\begin{aligned}w_2 &\leftarrow w_2 - \eta \frac{\partial \mathcal{L}}{\partial w_2} \\ &= w_2 - \eta \frac{\partial \mathcal{L}}{\partial f}\frac{\partial f}{\partial w_2} \\ &= 0 - 0.5(-9)(0.125) \\ &= 0.5625\end{aligned}$$

## Q1.2

Let $\mathbf{x} \in \mathbb{R}^N$ be the input vector, $\mathbf{W}_i \in \mathbb{R}^{M_{i+1} \times M_i}$ for $i = 1, ..., T$ be the $T$ hidden weights with $M_i \in \mathbb{N}$ and $M_1 = N, M_{T+1} = C$, and $\mathbf{b}_i \in \mathbb{R}^{M_{i+1}}$ be the hidden biases. Let $g(\mathbf{x}) = c\mathbf{x}$ for some $c \in \mathbb{R}$, since $g$ is linear. Hence with $\mathbf{h}_0 = \mathbf{x}$, for $i = 1, ..., T$ we have hidden outputs

$$\mathbf{h}_i = g(\mathbf{W}_i \mathbf{h}_{i-1} + \mathbf{b}_i) = c(\mathbf{W}_i \mathbf{h}_{i-1} + \mathbf{b}_i) \in \mathbb{R}^{M_{i+1}}$$

Rewriting this to a linear transformation and appending a 1 to the input for the bias trick,

$$\begin{bmatrix}\mathbf{h}_i \\ 1\end{bmatrix} = \mathbf{W}_i' \begin{bmatrix}\mathbf{h}_{i-1} \\ 1\end{bmatrix}$$

where $\mathbf{W}'_i = \begin{bmatrix} c\mathbf{W}_i & c\mathbf{b}_i \\ \mathbf{0} & 1 \end{bmatrix}$. That is, the output

$$\begin{bmatrix} \mathbf{y} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{h}_T \\ 1 \end{bmatrix} = \underbrace{\mathbf{W}'_T \dots \mathbf{W}'_1}_{\mathbf{W}} \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}$$

has effectively one collapsed parameter weight $\mathbf{W}$, which is sufficient to represent transformations across any $T$ hidden layers. So the number of hidden layers has no effect on the actual network.

## Q1.3

The gradient of the sigmoid function towards large $x$ (i.e. large error) saturates, meaning the parameters would not get updated significantly in that case. ReLU, on the other hand, does not saturate towards large $x$ so it is deemed a better choice.

## Q1.4

If we set the initial value of the network to be the same value, the back propagation update towards all parameters would be the same. It means essentially the network collapses to only one parameter and the model would not learn well.

## Q1.5

(a) AlexNet has in total $96\times(11\times11\times3+1)+256\times(5\times5\times48+1)+384\times(3\times3\times256+1)+384\times(3\times3\times192+1)+256\times(3\times3\times192+1)+(6\times6\times256+1)\times4096+(4096+1)\times4096+(4096+1)\times1000 = 34944+307456+885120+663936+442624+37752832+16781312+4097000 = \underline{60,965,224}$ parameters.

(b) VGG-16 has in total $64\times(3\times3\times3+1)+64\times(3\times3\times64+1)+128\times(3\times3\times64+1)+128\times(3\times3\times128+1)+256\times(3\times3\times128+1)+256\times(3\times3\times256+1)\times2+512\times(3\times3\times256+1)+512\times(3\times3\times512+1)\times2+512\times(3\times3\times512+1)\times3+(7\times7\times512+1)\times4096+(4096+1)\times4096+(4096+1)\times1000 = 1792+36928+73856+147584+295168+590080\times2+1180160+2359808\times2+2359808\times3+102764544+16781312+4097000 = \underline{138,357,544}$ parameters.

(c) From the paper[1], GoogLeNet has about in total $2.7\text{K}+112\text{K}+159\text{K}+380\text{K}+364\text{K}+437\text{K}+463\text{K}+580\text{K}+840\text{K}+1072\text{K}+1388\text{K}+1000\text{K} \simeq \underline{6,797,700}$ parameters.

[1] https://arxiv.org/pdf/1409.4842v1.pdf

From the calculations, we see GoogLeNet has the fewest number of paramters and VGG-16 has the most. While GoogLeNet is the deepest, it does not use any fully connected layers, which contribute to most of the number of parameters in AlexNet and VGG-16.

# 4 Training

## Q4.1

This is the training log:

```
cost = 0.273491 training_percent = 0.910000
cost = 0.279565 training_percent = 0.910000
cost = 0.176619 training_percent = 0.920000
cost = 0.127344 training_percent = 0.950000
cost = 0.191895 training_percent = 0.960000
test accuracy:  0.944000

cost = 0.192910 training_percent = 0.930000
cost = 0.131836 training_percent = 0.970000
cost = 0.115812 training_percent = 0.970000
cost = 0.103636 training_percent = 0.970000
cost = 0.124224 training_percent = 0.980000
test accuracy:  0.960000

cost = 0.111115 training_percent = 0.960000
cost = 0.113216 training_percent = 0.940000
cost = 0.134874 training_percent = 0.960000
cost = 0.067548 training_percent = 0.990000
cost = 0.095426 training_percent = 0.980000
test accuracy:  0.966000

cost = 0.086685 training_percent = 0.980000
cost = 0.106186 training_percent = 0.950000
cost = 0.034245 training_percent = 1.000000
cost = 0.048397 training_percent = 1.000000
cost = 0.060728 training_percent = 0.970000
test accuracy:  0.968000

cost = 0.069977 training_percent = 1.000000
cost = 0.068312 training_percent = 0.980000
cost = 0.063643 training_percent = 0.980000
cost = 0.084625 training_percent = 0.960000
```

```
cost = 0.083214 training_percent = 0.980000
test accuracy:  0.970000

cost = 0.083081 training_percent = 0.970000
cost = 0.026531 training_percent = 1.000000
cost = 0.044653 training_percent = 0.980000
cost = 0.056298 training_percent = 0.980000
cost = 0.049833 training_percent = 0.990000
test accuracy:  0.970000
```

The final test accuracy is 97%.

### Q4.2

The confusion matrix is

$$\begin{bmatrix} 56 & 0 & 2 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 53 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 39 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 1 & 53 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 42 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 43 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 42 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 48 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 2 & 0 & 0 & 52 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 56 \end{bmatrix}.$$

There are multiple ties when it comes to the most confused pairs. Class 3 has two misclassified as class 1, class 6 has 2 misclassified as class 4 and 2 as class 9, class 7 has 2 misclassified as class 1 and class 8 has 2 misclassified as class 3. Some of them are reasonable, for example the character (when written poorly), '6' (class 7) can be similar to '0' (class 1), and '5' (class 6) can be similar to '3' (class 4). The other misclassification may also be because of poor writing.

## 5 Visualization

### Q5.1

The plots of layer 2 and layer 3 are exactly the same because layer 3 is ReLU, which clamps negative values to zero, which is also done in the `imshow` function since it cannot visualise negative pixel values.

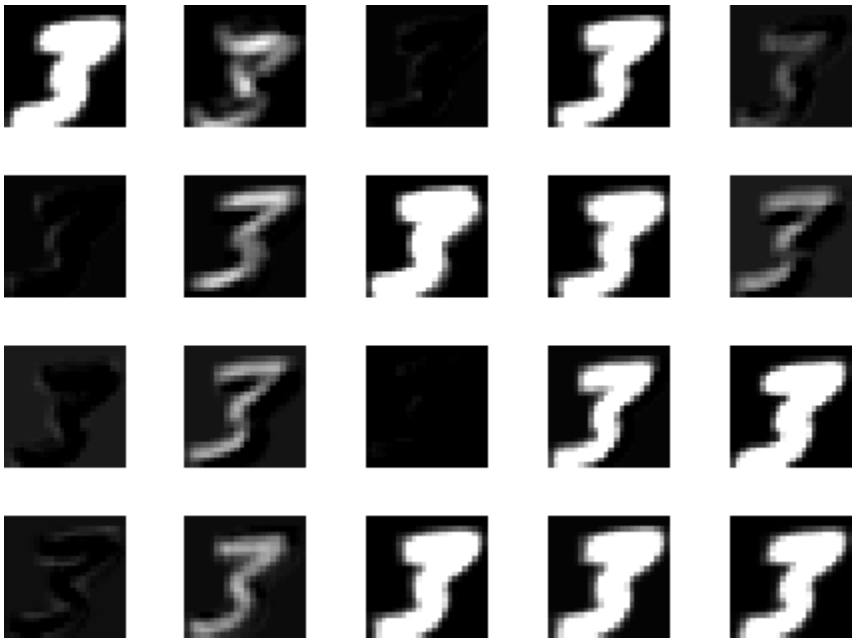

Figure 1: Output of layer 2 (negative pixel values clamped to 0).

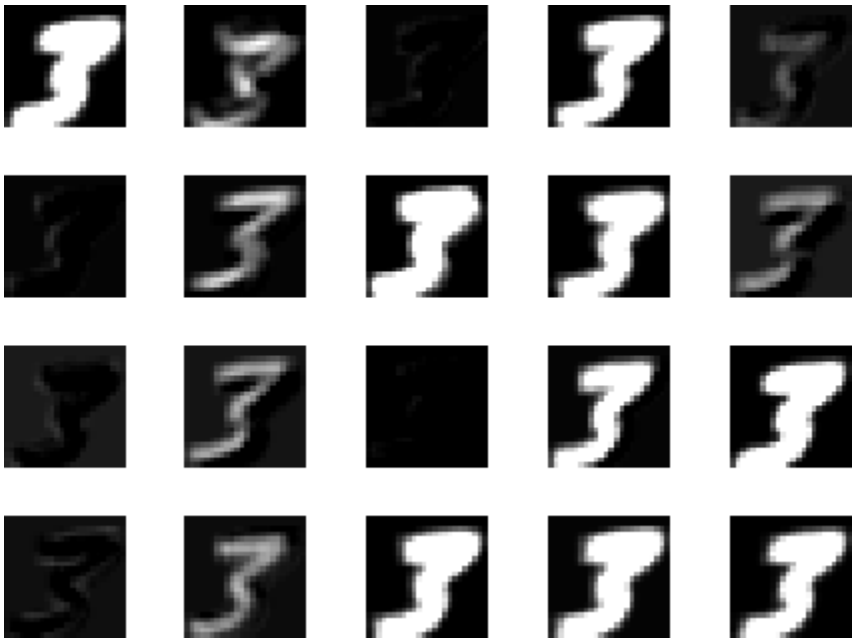

Figure 2: Output of layer 3.

## Q5.2

The original digit is shown below.

Figure 3: Original image.

Most of feature images look quite similar to the original image. The differences are that

1. Some feature maps are completely dark.
2. Some feature maps are like applying erosion to the original image, i.e. it looks thinner than the original.
3. Some feature maps are like applying dilation to the original image, i.e. it looks fatter than the original.

# 6 Image Classification

Figure 4: Prediction on image 1, 100% correct.

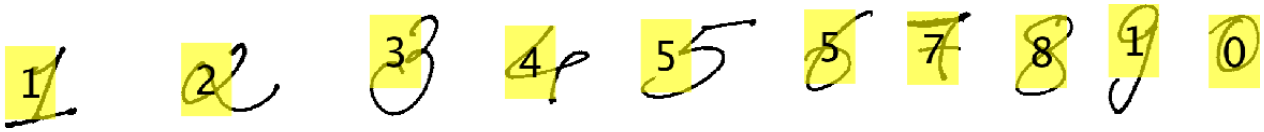

Figure 5: Prediction on image 2, 80% correct.

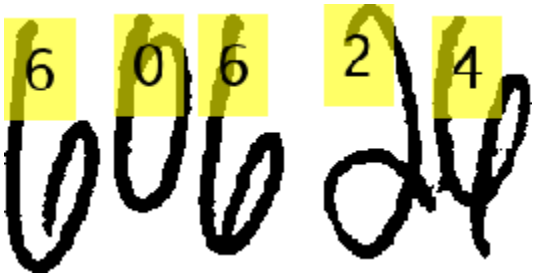

Figure 6: Prediction on image 3, 100% correct.

Figure 7: Prediction on image 4, 84% correct.