zheyaoz HW

Zheyao Zhu

March 2019

Q1.1

$softmax(x_i + c) = \frac{e^{x_i + c}}{\sum_j e^{x_j + c}} = \frac{e^c e^{x_i}}{e^c \sum_j e^{x_j}} = \frac{e^{x_i}}{\sum_j e^{x_j}} = softmax(x_i)$

Q1.2

1) The range of elements after softmax is applied will be in the interval [0,1]
2) Probability distribution
3) The first step is applying the standard exponential distribution to each element of the vector. The second and the third step are normalizing the vector.

Q1.3

Let's suppose there s a neural net with two hidden layers and a linear activation function

$$\begin{aligned} y &= C(h_2W_2 + b_3) \\ &= C(C(h_1W_2W_3 + b_2W_3) + b_3) \\ &= C(C(C(xW_1 + b_1)W_2W_3 + b_2W_3) + b_3)) \\ &= C((C^2xW_1W_2W_3 + C^2b_1W_2W_3 + C^2b_2W_3) + b_3) \\ &= C^3xW_1W_2W_3 + C^3b_1W_2W_3 + C^2b_2W_3 + Cb_3 \\ &= C(W'x + b') \end{aligned}$$

We can see that a multi-layer neural that has a linear activation function behaves the same as a single layer neural network which is equivalent to linear regression.

Q1.4

$$\sigma(x) = \frac{1}{1+e^{-x}} = (1+e^{-x})^{-1}$$

To compute the gradient of the activation function, we take the partial derivative of the function with respect to x:

$$\frac{\partial \sigma}{\partial x} = (1+e^{-x})^{-2} * e^{-x}$$
$$= (\frac{1}{1+e^{-x}})^2 * (e^{-x} + 1 - 1)$$
$$= \sigma(x)^2 * (\frac{1}{\sigma(x)} - 1)$$

Q1.5

We know that $y = x^T W + b$ and that $\frac{\partial J}{\partial y} = \delta$

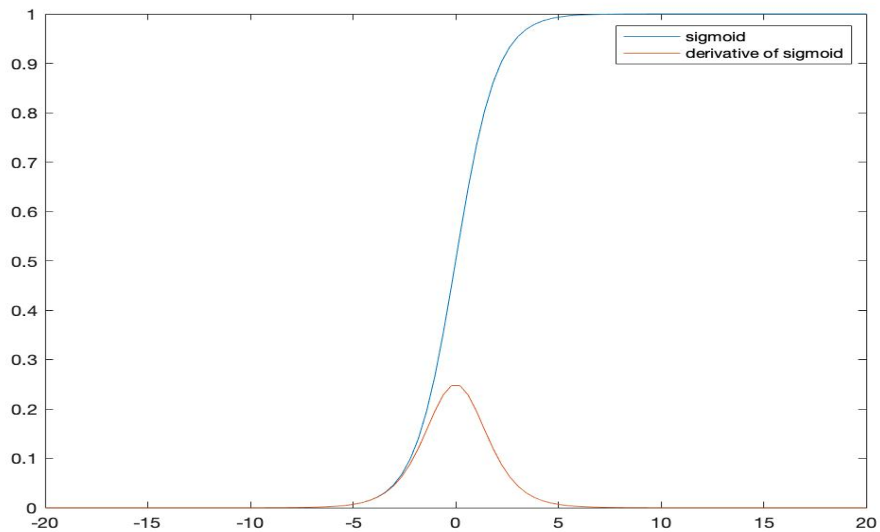
$\frac{\partial y}{\partial w} = x^T$ $\quad \frac{\partial J}{\partial w} = \frac{\partial J}{\partial y}\frac{\partial y}{\partial w} = \delta x^T$

$\frac{\partial y}{\partial x} = W$ $\quad \frac{\partial J}{\partial x} = \frac{\partial J}{\partial y}\frac{\partial y}{\partial x} = \delta W$

$\frac{\partial y}{\partial b} = 1$ $\quad \frac{\partial J}{\partial b} = \frac{\partial J}{\partial y}\frac{\partial y}{\partial b} = \delta$

Q1.6

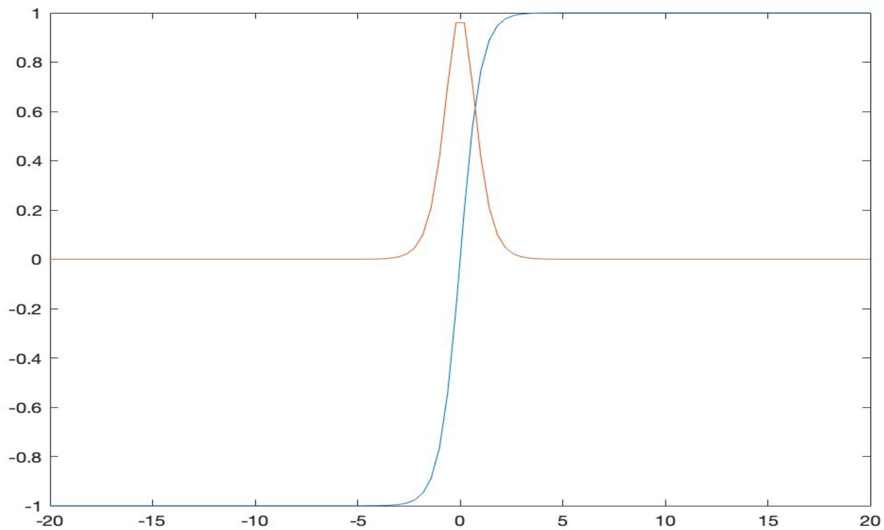
1.



plot of sigmoid and it's derivative

As we can see that the range of sigmoid is [0,1], but the range of the derivative of sigmoid is only [0,0.25]. Since the derivative of the activation function scales each layer of the back propogation, the gradients of of a multi-layer neural network would eventually vanish, as the gradient of each layer gets cumulatively scaled by a number less than 0.25.

2).
Output range of sigmoid is [0,1]
Output range of tanh(x) is [-1,1]



plot of tanhx

We would prefer tanhx because as we can see in the graph, the range of the derivative of tanhx is larger. The range is [0,1] which is the same as the output of sigmoid.

3).
As mentioned in the previous question, the range of output of the derivative of tanhx is [0,1] which is 4 times larger than the range of derivative of sigmoid. Thus there will be less of a vanishing gradient issue.

4).

$$\begin{aligned} tanh(x) &= \frac{1-e^{-2x}}{1+e^{-2x}} \\ &= \frac{2-1-e^{-2x}}{1+e^{-2x}} \\ &= \frac{2}{1+e^{-2x}} - 1 \\ &= 2\sigma(2x) - 1 \end{aligned}$$

Q2.1.1

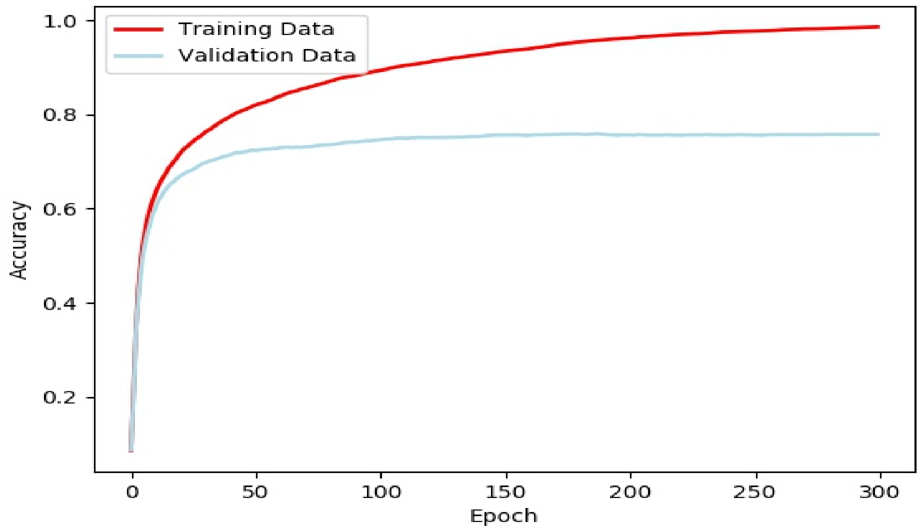
The reason that it's not a good idea to initialize a network with all zero is that, in each iteration of the back-propagation algorithm, the weights are updated by multiplying the existing weight by a delta determined by the algorithm. If the initial weight value is 0, then multiplying it by any delta value won't change the weight which means each iteration has no effect on the weights you're trying to optimize. The traning process would essentially be stuck.

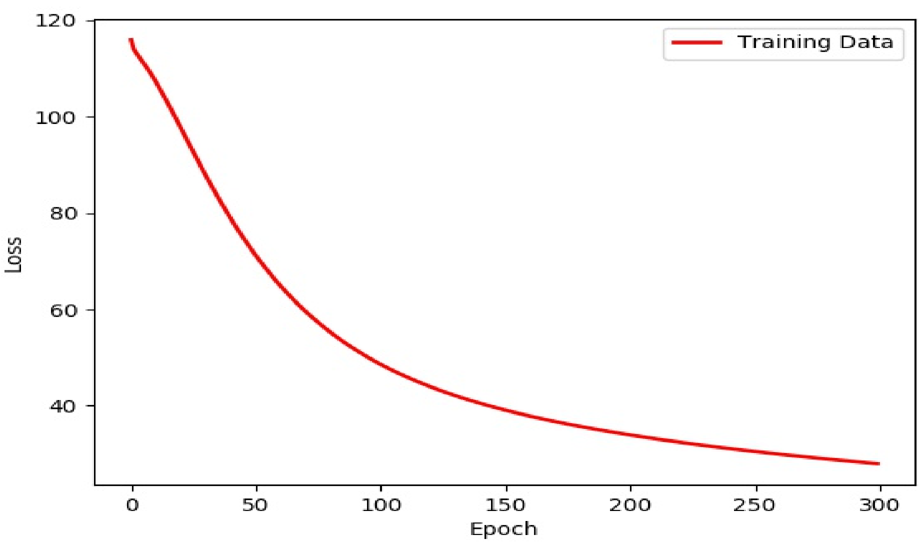
Q2.1.3

The purpose of initializing the neural network with random weights is so that we don't have a symmetric neural network in which case, different node of the neural network would be exactly same.

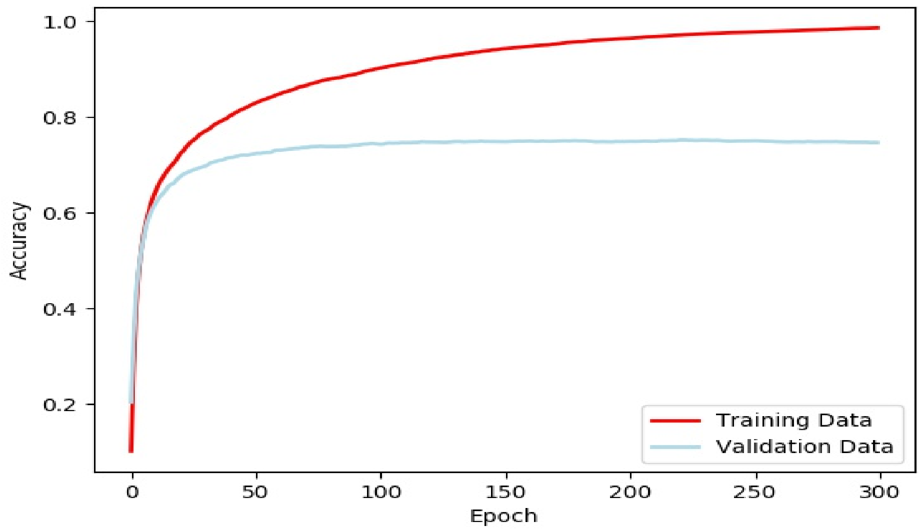
Q3.1.2

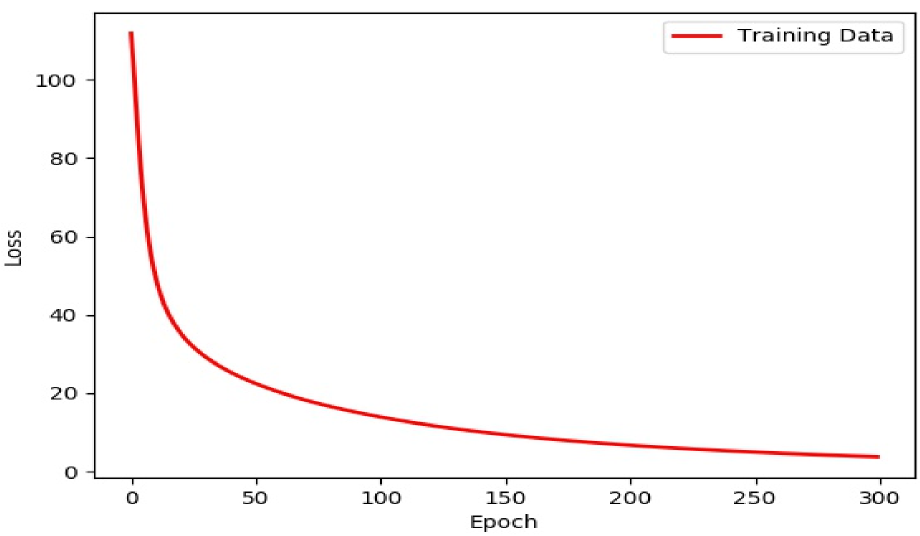
One-tenth learning rate : $1 * 10^{-4}$



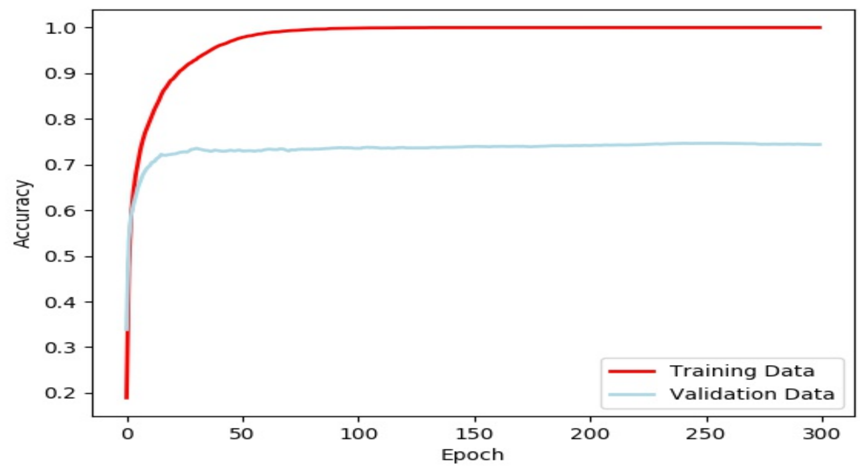


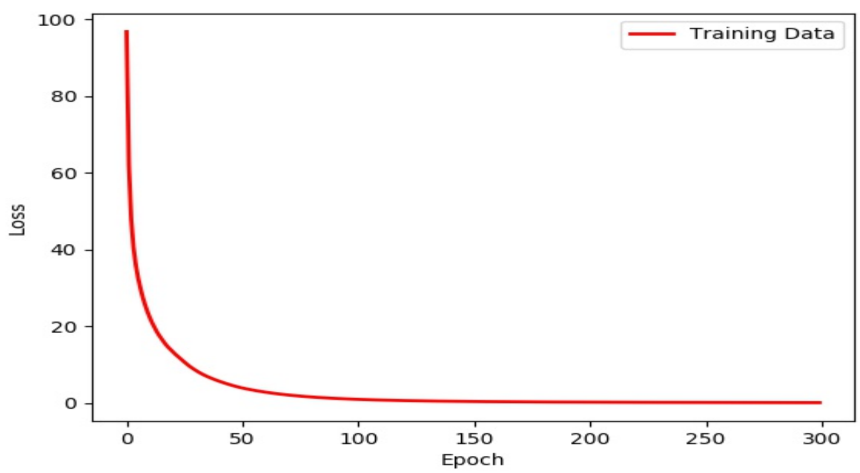
Regular learning rate : $1 * 10^{-3}$





Ten times learning rate : $1 * 10^{-2}$



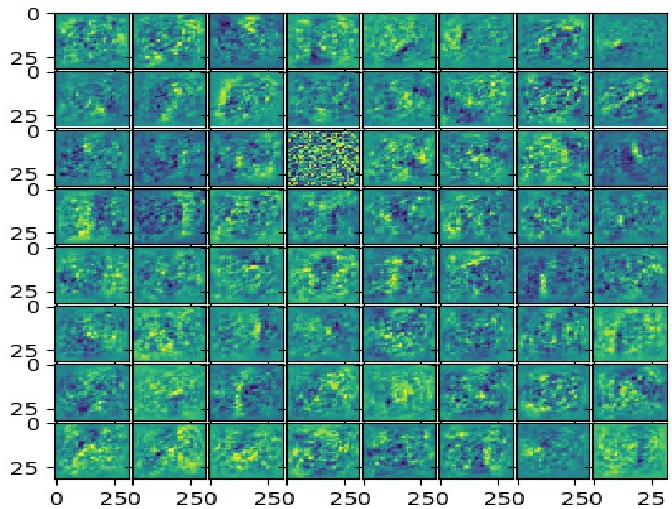


Comment: From the above images, learning rate has a big affect on how fast the loss drops down during the training process. When learning rate is high, loss drops down a lot faster. Higher learning rate also makes training accuracy improve faster.

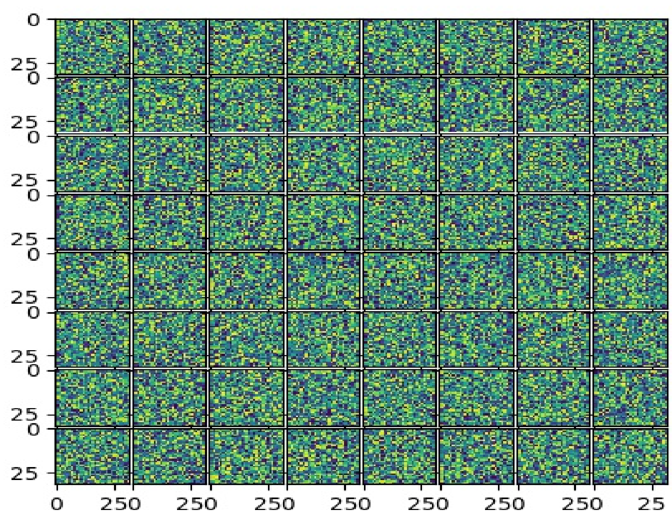
Final Validation Accuracy of the Best Network : 76%

Q3.1.3

Trained Weight

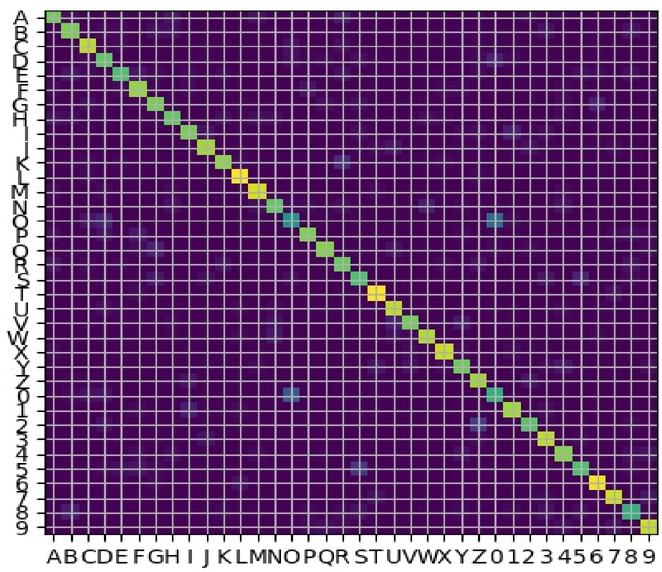


Initialized weight:



Q3.1.4

Confusion Matrix:

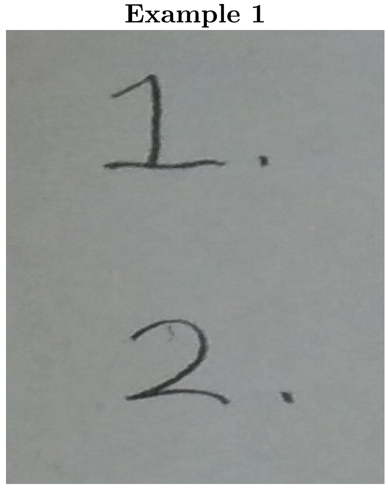


Q4.1

Assumption 1: All connected groups of pixels are characters.

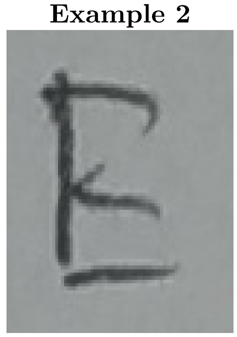
Assumption 2: There are no broken lines within each hand-written character

Example 1



Using the algorithm described in the handout, these dots next to the numbers would be identified as characters. I solved this problem by only taking connected groups of pixels that are larger than certain threshold.

Example 2



In this image, we can see that the bottom line of the character E is actually disconnnected with the rest of the character. So when labeling the image, the bottom line would be left outl from the rest of the character

Q4.3

Image 1

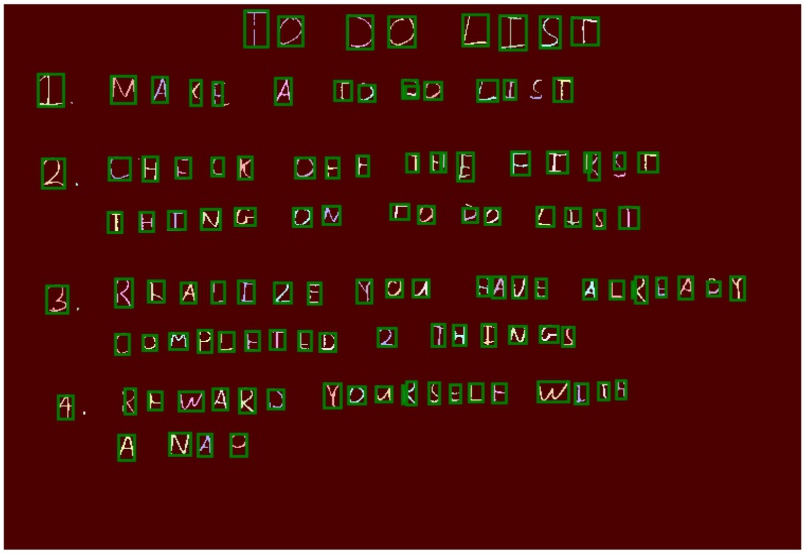


Image 2

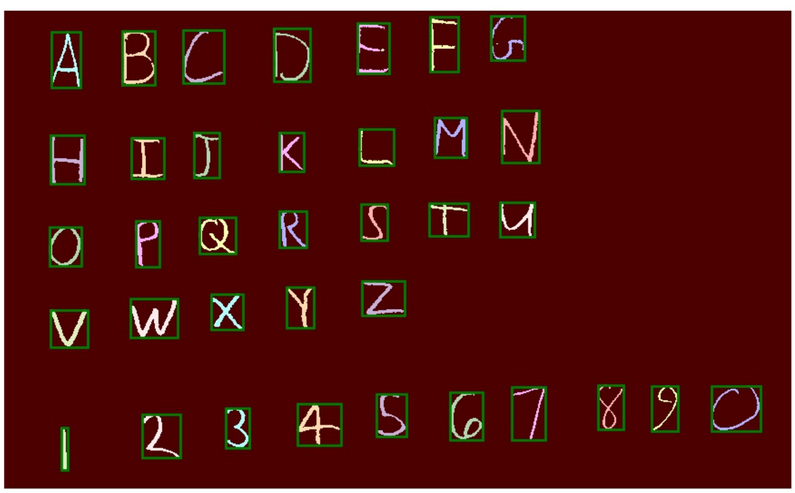
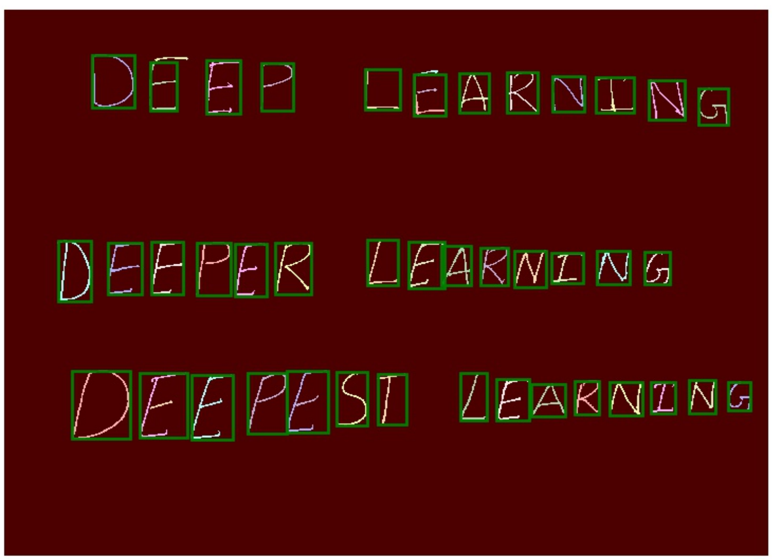


Image 3

Image 4



Q4.4

Predicted Result of Image 1

```
01_list.jpg
F 5 J F C I 5 T

I M A Y F A T V F Y V 6 I T

I C H F F K U F F Y F 1 V F T 5 Y T T

Y Y F N V V Y T O Y O C F J T

B X F A V T J F Y O U Y A V F X V 1 Y F F Y Y

F V Y Y V F Y F C 2 Y H V N F S

Y I F V 1 Y V Y O U X 5 S F V F V T Y Y

A N A R
```

Predicted Result of Image 2

```
02_letters.jpg
U B C J H F 5

H I 1 K L M N

0 P Q K S T V

V W X Y Z

1 Z 3 F S G J V 7 0
```

Predicted Result of Image 3

Predicted Result of Image 4