Abhishek Bamotra
Andrew ID: abamotra

Homework 5

Problem 1

Q 1.1

Softmax is defined as below, for each index i in a vector x.

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

Therefore,

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

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

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

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

The value of c should be as - max x_i because that will make sure that the coefficient of the exponential term is always greater than 0. Because of which the numerator will be always greater than 1. This scenario will avoid any divisions between extremely small numbers. This technique will therefore avoid such errors and make the function stable.

Q 1.2

- The range of each element is **(0, 1]**. The sum over all elements is **1**.
- The softmax takes an arbitrary real valued vector x and turns it into the probability distribution; **softmax(item)**, softmax of an item gives us the probability of choosing that item among all other items.
- **Step 1:** Makes each dimension of the vector positive **Step 2:** Sums up each of the dimension **Step 3:** Makes it a probability distribution.

Q 1.3

We know the equation of a linear regression is:

$$y = W^T X + b$$

Let's consider two linear layers without a non-linear activation function, we have:

$$y_2 = W_2^T (W_1^T X + b_1) + b_2$$

$$y_2 = W_2 W_1^T X + W_2 b_1 + b_2 \tag{1}$$

From equation (1), we can say it is a linear regression equation with weight equal to $W_2 W_1^T$ and bias equal to $W_2 b_1 + b_2$. This example for two layers can be repeated for multi-layer network.

Q 1.4

Sigmoid function is defined as:

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

Taking derivative of sigmoid function ($\sigma(x)$), we get:

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

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

From above, we can see we can find derivative of sigmoid function without having access to x directly.

Q 1.5

We know,

$$y = Wx + b \qquad or \qquad y_i = \sum_{j=1}^{d} x_j W_{ij} + b_i \tag{2}$$

$$\frac{\partial J}{\partial y} = \delta \in \mathbb{R}^{k \times 1} \tag{3}$$

If we differentiate y_i from equation (2) w.r.t W_{ij}, x_j and b_i, we get:

$$\frac{\partial y_i}{\partial W_{ij}} = x_j$$

$$\frac{\partial y_i}{\partial x_j} = W_i j$$

$$\frac{\partial y_i}{\partial b_i} = 1$$

Using the above derivatives, we can find $\frac{\partial J}{\partial W}$, $\frac{\partial J}{\partial x}$ and $\frac{\partial J}{\partial b}$ as:

$$\frac{\partial J}{\partial W} = \sum_{i=1}^{k} \frac{\partial J}{\partial y_i} \frac{\partial y_i}{\partial W}$$

$$\frac{\partial J}{\partial x} = \sum_{i=1}^{k} \frac{\partial J}{\partial y_i} \frac{\partial y_i}{\partial x}$$

$$\frac{\partial J}{\partial b} = \sum_{i=1}^{k} \frac{\partial J}{\partial y_i} \frac{\partial y_i}{\partial b}$$

Converting it to matrix form, we get:

$$\frac{\partial J}{\partial W} = x\delta^T \in \mathbb{R}^{d \times k}$$

$$\frac{\partial J}{\partial x} = W\delta \in \mathbb{R}^{d \times 1}$$

$$\frac{\partial J}{\partial b} = \delta \in \mathbb{R}^{k \times 1}$$

Q 1.6

1

Sigmoid function maps input space $\mathbb{R}$ to [0, 1], i.e it maps a huge range of space to a constraint space of [0, 1] which is very small compared to $\mathbb{R}$. If the take the gradient of sigmoid value in range [0, 1], it is in the range of [0, 0.25]. Therefore, if we propagate through multiple layers, the gradient will "Vanish" i.e become almost zero. Because of which there will be no change in parameters of the network and hence, difficulty in learning the model.

2

The output range of tanh is [-1, 1] and sigmoid is [0, 1]. We prefer tanh over sigmoid because of larger output range. Also, it has less of a vanishing gradient problem.

3

tanh(x) has less of a vanishing gradient problem as it's output range [-1, 1] is larger than that of sigmoid [0, 1]. and the derivative of tanh(x) is in the range of [0, 1]. The larger derivative range for tanh(x) makes the derivative for tanh(x) stronger and larger than sigmoid derivative.

4

We know,

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

Problem 2

Q 2.1

Q 2.1.1

It is not a good idea to initialize a network with all zeros because if we do so, the gradient of all the weights will point in a single direction for every iteration. Such initialization can result in same set of weights corresponding to all the output nodes. Example, for a given input, output nodes show equal values. Also, because of single direction of all gradient of weights will result in higher convergence time.

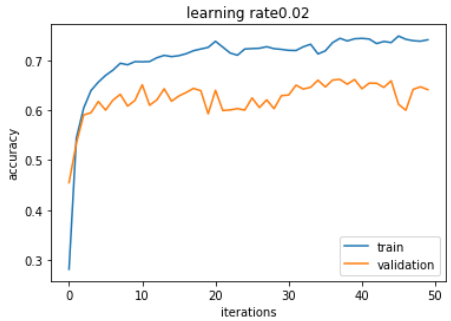
Q 2.1.3

We initialize weights with random numbers to avoid the symmetry we achieve due to zero initialization of the weights as discussed in Q 2.1.1 above.

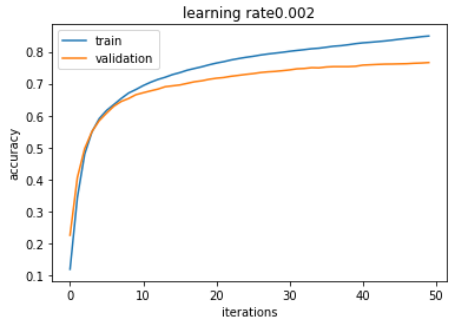
We scale the initialization depending on layer size because if we initialize weights for the deeper layers with values close to zero, it will result in very low value of gradients at initial layers of the network during backward propagation. To avoid this, we scale the initial weights depending on the layer size i.e giving higher initial weights for deeper layers.

Problem 3

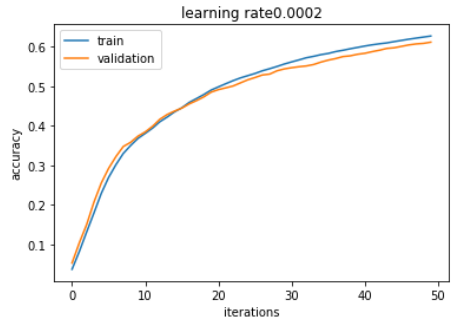
Q 3.2



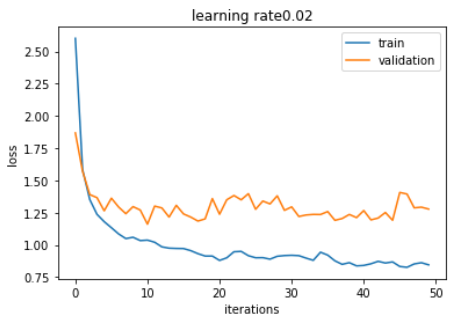
(a) Learning rate 0.02



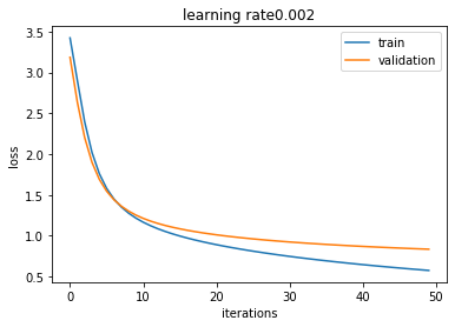
(b) Learning rate 0.002 (Best)



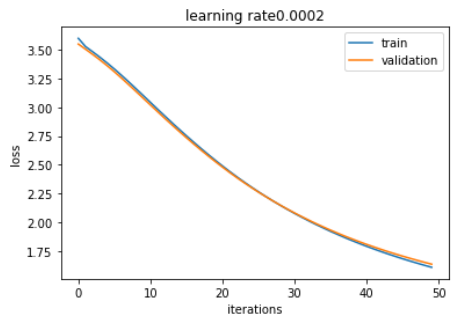
(c) Learning rate 0.0002



(d) Learning rate 0.02



(e) Learning rate 0.002 (Best)



(f) Learning rate 0.0002

Figure 1: Accuracy vs Iteration and Loss vs Iteration plots for best learning rate along with 10 times and 1/10th of the best learning rate

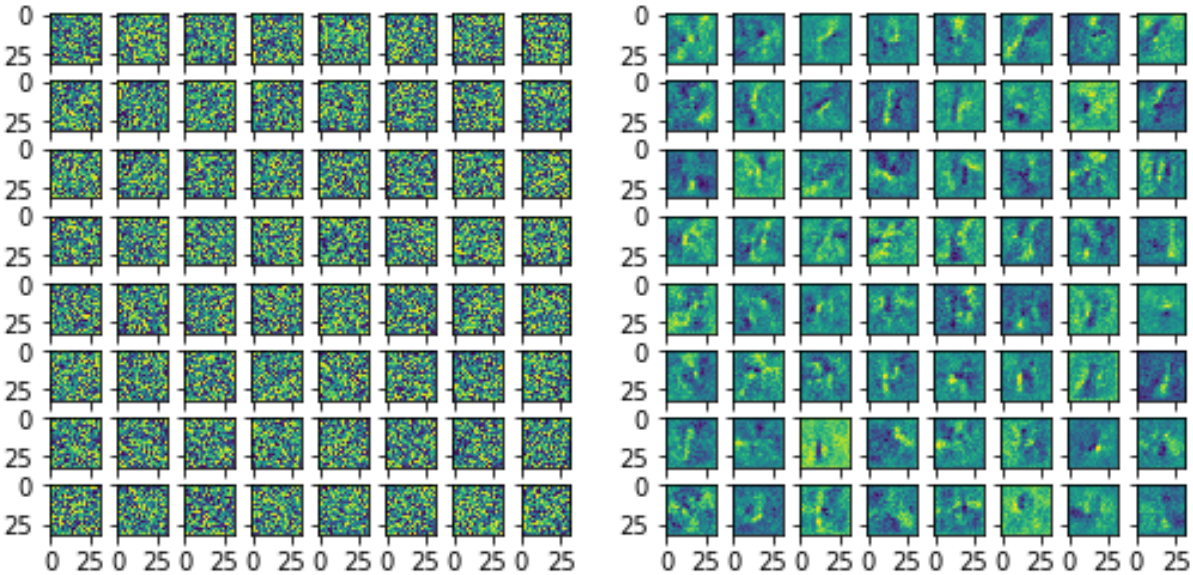
The accuracy and loss curve with iteration for best learning rate of 0.002 are shown in Figure (1) b and e. Also, curves for 10 times and 1/10th of the best learning rate is also shown in Figure (1).

From the plots Figure 1a and 1d, we can say that with a bigger learning rate (i.e 0.02), the loss is fluctuating between the optimal solution and therefore giving jittery curve. Because of which the accuracy is also being fluctuated.

From the plots Figure 1c and 1f, we can say that with smaller learning rate (i.e 0.0002), the loss curve is very smooth but it never reaches optimal solution in same number of iterations. This is because the update step size is very small and takes much more iterations to converge to the optimal solution.

For best learning rate of 0.002, I'm able to get **validation accuracy of 76.64%** and **test accuracy of 77.55%**.

Q 3.3



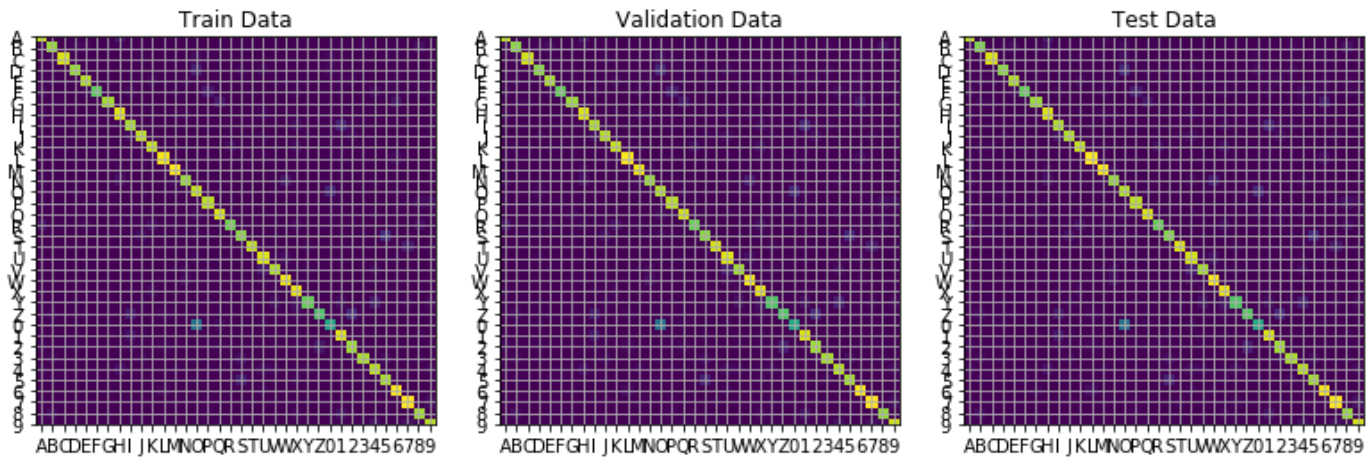
(a) Initialized weights

(b) Immediately after Initialization

Figure 2: Visualization of network weights

The initialized and immediately after initialization weights are visualized in the Figure 2. In the initialized weights visualization, we can see that the weights are randomly initialized whereas in the after initialized weights, we can see patterns appearing in the visualization. These patterns in the visualization mean that the network is trying to learn patterns like edges of characters and digits of various orientations and scales.

Q 3.4



(a) Train Split (b) Validation Split (c) Test Split

Figure 3: Confusion matrices for training, validataion and test data set.

Confusion matrices for training, validation and test data set are shown in the Figure (3).

From the confusion matrices above and multiple runs of the code, we can clearly say that the most confused pair of classes are **'O' and '0' (zero)**.

Second most confused pair of classes are **'S' and '5'**.

Third and fourth most confused pairs of classes are **'I' and '1'** and **'Z' and '2'**.

Problem 4

Q 4.1

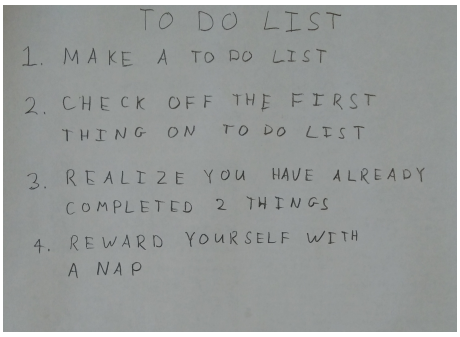
(a) Connected characters (b) Some small characters

Figure 4: Two big assumptions that the sample method makes

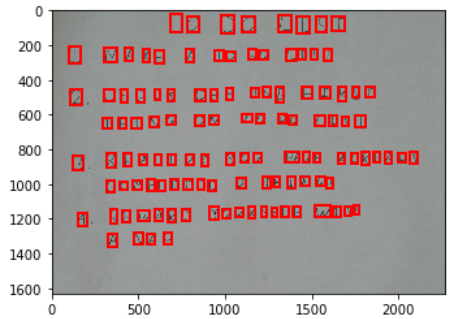
Sample method makes two big assumptions i.e all the characters are separate and not connected. So, if any two characters are connected, it fails.

Second big assumption is the characters are of similar size. So, we are using morphology processing to remove noise which will consider any small character or mark compared to the big characters as noise and remove it. Because of which if there are characters of very different sizes as shown in Figure 4b, it will remove the smaller characters considering those as noise.

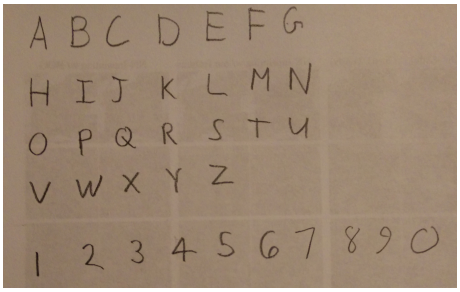
Q 4.3



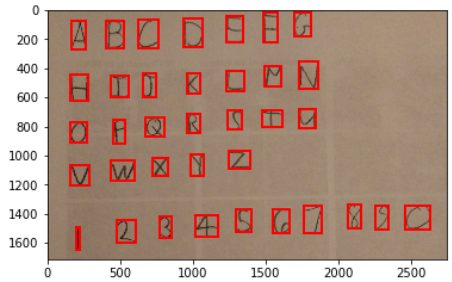
(a) Original Image 1



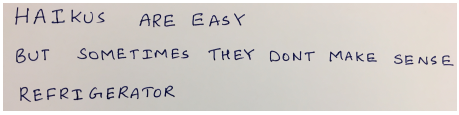
(b) Image with located boxes 1



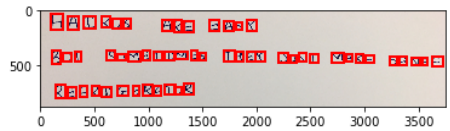
(c) Original Image 2



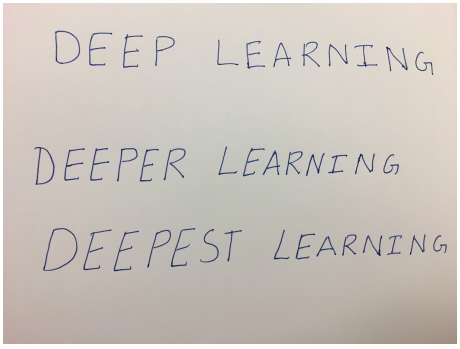
(d) Image with located boxes 2



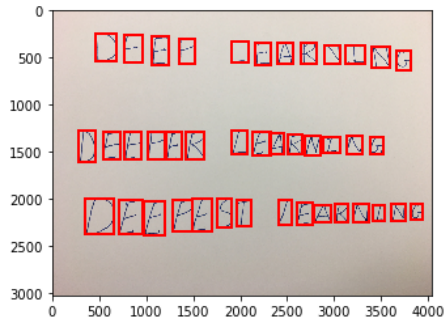
(e) Original Image 3



(f) Image with located boxes 3



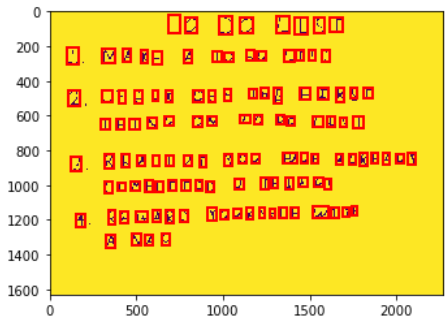
(g) Original Image 4



(h) Image with located boxes 4

Figure 5: Images with located boxes using findLetters()

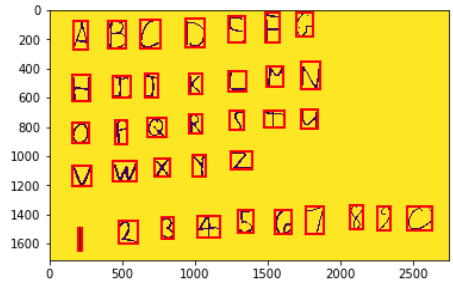
Q 4.4



(a) Image with located box

```
TQDQLIST
INAKEATQQQLIST
2CHKCKDFET7HEDFIRFT
THIWGQWQQLIST
3RXALIZEYOUTHHIWVEGSALR2ADY
CQMPLXTLDZ
4RXWARDYQWRSELFWITH
ANAP
```

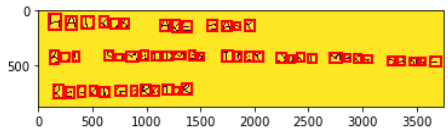
(b) Output text



(c) Image with located box

```
ABCDGFG
HIIKLMN
QPQRSTW
VWXYZ
1Z3GSG789S
```

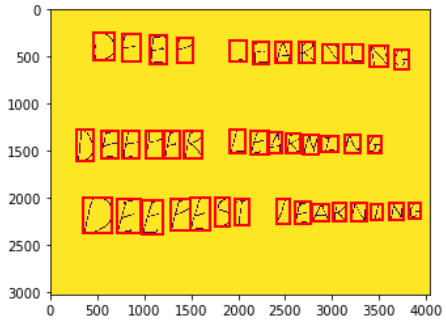
(d) Output text



(e) Image with located box

```
HAIKUSAREHAGY
BLTSDMETIMBSTHEXDOWTMAK6SEKGG
RBFRIGERATQR
```

(f) Output text



(g) Image with located box

```
DYEPLEARMING
DEE9EKLEARNIMG
DE8PESILEARMIMG
```

(h) Output text

Figure 6: Images with located boxes and corresponding text output

Problem 5

Q 5.2

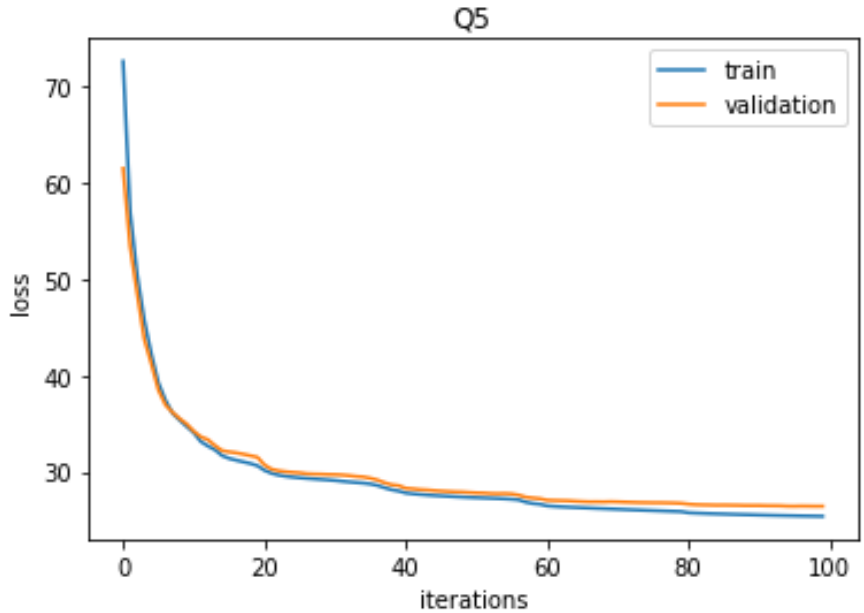


Figure 7: Training loss curve

We can see from the loss curve that the loss drops drastically in the initial iterations as the learning rate is high in the beginning and momentum of convergence decreases as the number of iterations increase as the learning rate gets smaller as number of iterations increase.

Q 5.3.1

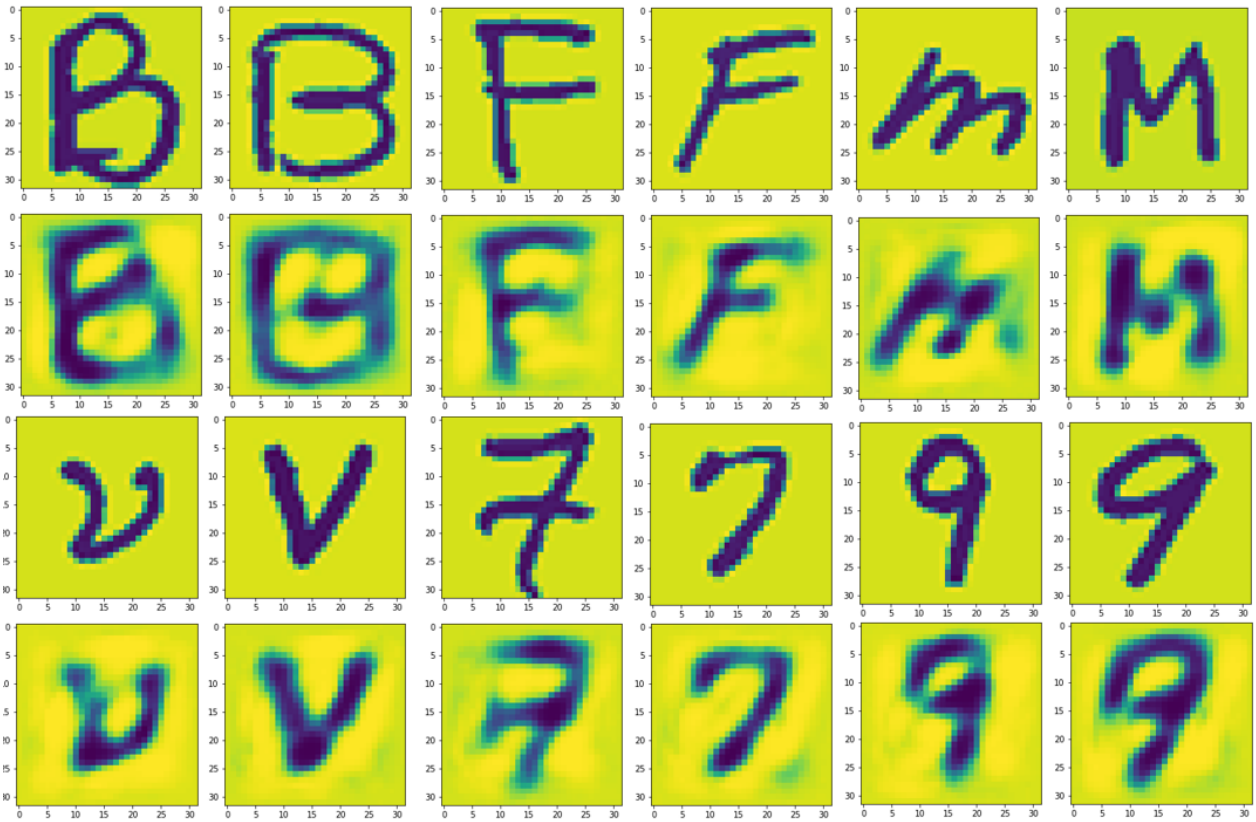


Figure 8: Original Images with reconstructed images

Six different classes are shown in Figure 8 with original images in row 1 and 3 with reconstructed images in row 2 and 4 of the figure. We can see from the results that the reconstructed images are very similar to the original images. The reconstructed images are bleeding at the edges and are blurred as compared to the original images. But, the reconstruction is pretty good.

Q 5.3.2

Average PSNR value from the autoencoder across all images in the validation set is **16.32037072440959**.

Problem 6

Q 6.1.1

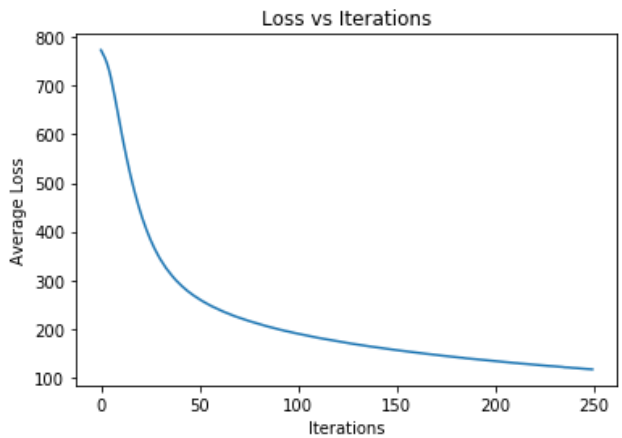


Figure 9: Loss curve for fully connected

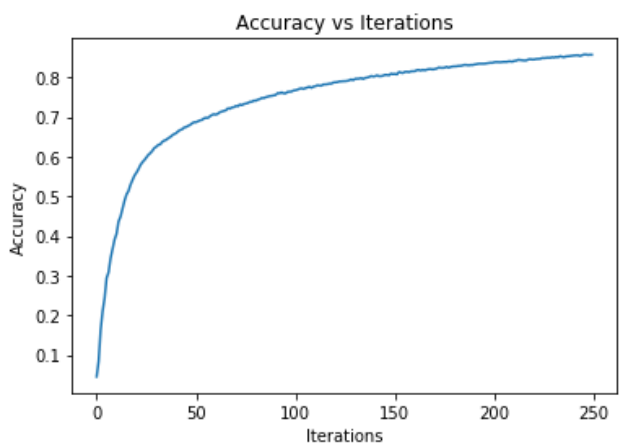


Figure 10: Accuracy curve for fully connected

Final test accuracy, I got is 78.39% and final train accuracy is 85.65% for number of iterations as 250.

Q 6.1.2

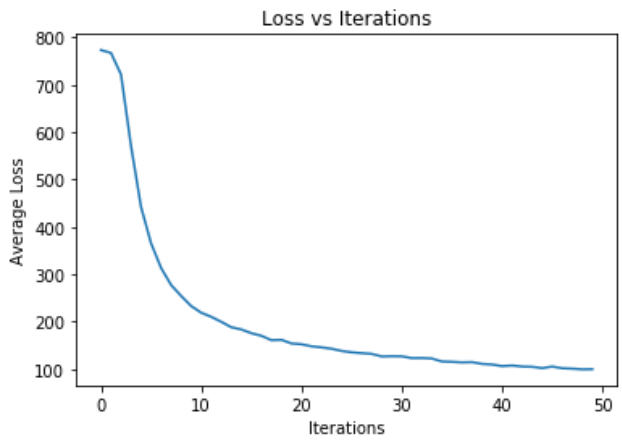


Figure 11: Loss curve for Conv

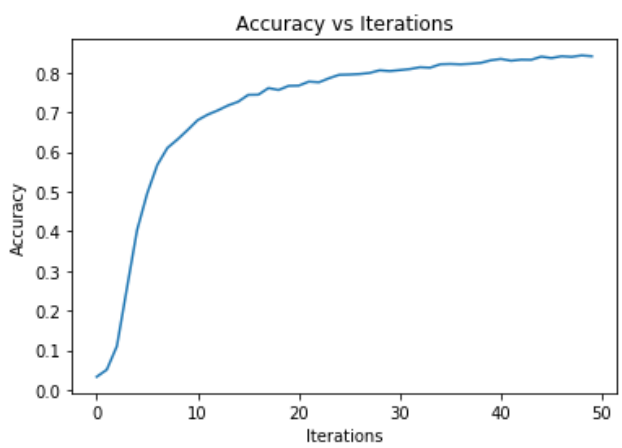


Figure 12: Accuracy curve for Conv

Final test accuracy, I got is 91.17% with number of iterations as 50. From the results, we can say that convolutional neural network is much faster in convergering as compared to the fully connected network. The drop in loss is quiet high in conv neural network.

Q 6.1.3

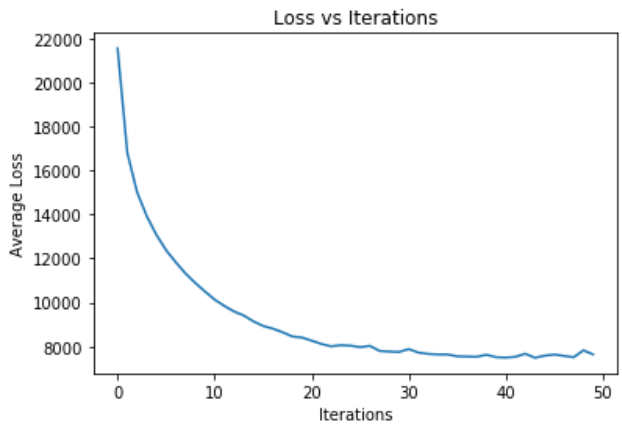


Figure 13: Loss curve for CIFAR-10

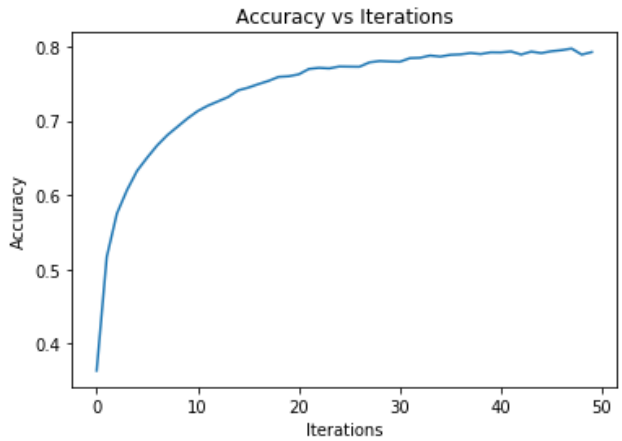


Figure 14: Accuracy curve for CIFAR-10

Final test accuracy, I got is 57.53% and final train accuracy is 79.25% for number of iterations as 250.

Q 6.2

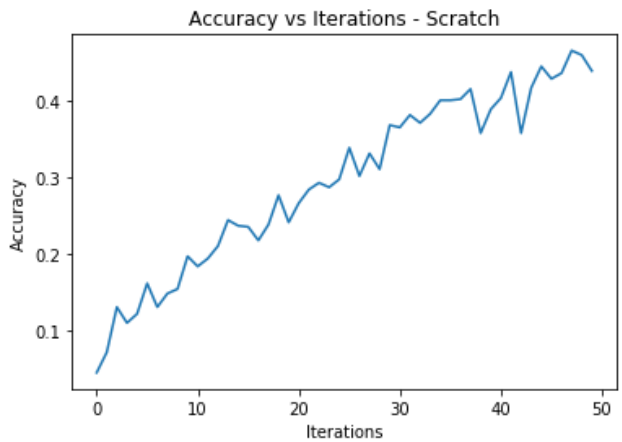


Figure 15: Accuracy curve for Scratch network

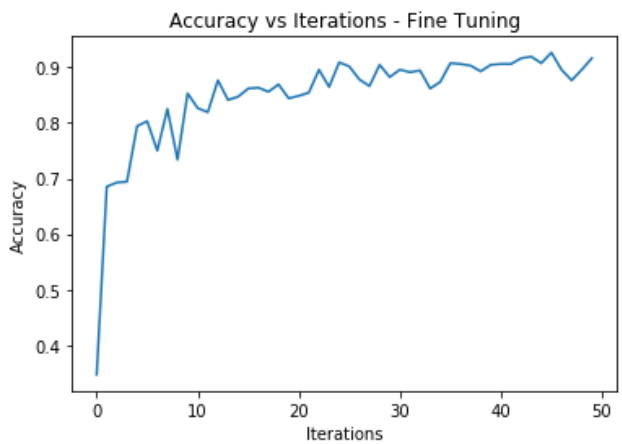


Figure 16: Accuracy curve for squeezenet1_1

For the scratch network, I'm able to get an accuracy of:Test Accuracy: 36.47%, Train Accuracy: 43.82%

Whereas for sqeezenet1_1, I'm able to get an accuracy of: Test Accuracy: 81.17%, Train Accuracy: 90.88%

From the above results, we can clearly see for the same number of iterations, squeezenet out performs scratch network by a huge difference. When the model weights are fine-tuned with the initialized weight from pretrained models, it is able to give a high accuracy of 80.29% on the test set and 91.61% on train set. Therefore, the results from squeezenet are promising.