16-720A Computer Vision: Homework 5 Neural Networks for Recognition

Radhika Mohanan(rmohanan)

1 Theory

Q1.1

For an index i in a vector x, we have:

$$\begin{aligned} \operatorname{softmax}(x_i) &= \frac{e^{x_i}}{\sum_j e^{x_j}} \\ \operatorname{softmax}(x_i + c) &= \frac{e^{x_i+c}}{\sum_j e^{x_j+c}} \\ \operatorname{softmax}(x_i + c) &= \frac{e^{x_i} e^{c}}{e^{c} \sum_j e^{x_j}} \end{aligned} \tag{1}$$

e^c factored out from both the numerator and the denominator can be cancelled. Giving us the equation as below:

$$\begin{aligned} \operatorname{softmax}(x_i + c) &= \frac{e^{x_i}}{\sum_j e^{x_j}} \\ \therefore \operatorname{softmax}(x_i) &= \operatorname{softmax}(x_i + c) \end{aligned} \tag{2}$$

When we set $c = -\max x_i$ or subtract the maximum value exponent term, we get $e^0 = 1$ and all the other terms will be scaled between 0 and 1, thus avoiding large exponent terms which improves numerical stability and avoids overflows.

Q1.2

Q1.2.1

For every element when $d > 2$, the interval will be (0, 1) and the sum over all elements will be equal to 1.

Q1.2.2

One could say that *"Softmax takes an arbitrary real valued vector x and turns it into a vector of size (d, 1) with each element of the vector in the interval (0, 1) and sum over all elements equal to 1 (probability distribution)."*

Q1.2.3

$s_i = e^{x_i}$ calculates the outcome frequency x_i in the exponential form. $S = \sum s_i$ calculates the total outcome frequency.$\frac{s_i}{S}$ normalize the frequency of each x_i and outputs the probability.

Q1.3

A network without non-linear activation functions will have the x value change according to linear function : $x_{i+1} = W_i x_i + b_i$
When applying to multi-layer neural networks, we have:

$$\begin{aligned}
y &= W_n x_n + b_n && (3)\\
&= W_n(W_{n-1}x_{n-1} + b_{n-1}) + b_n && (4)\\
&= W_n W_{n-1} x_{n-1} + W_n b_{n-1} + b_n && (5)\\
&= W^{'} x_{n-} + b^{'} && (6)\\
&= W^{'}(W_{n-2}x_{n-2} + b_{n-2}) + b^{'} && (7)\\
&\text{............} && (8)\\
&= Wx + b && (9)
\end{aligned}$$

which is the same as solving a linear regression problem

Q1.4

Given sigmoid activation function is :

$$\sigma(x) = \frac{1}{1+e^{-x}} = \left(1+e^{-x}\right)^{-1} \tag{10}$$

Differentiating the above equation with respect to x using with chain rule we get:

$$\begin{aligned}\frac{d}{dx}\sigma(x) &= \frac{d}{dx}\left(\left(1+e^{-x}\right)^{-1}\right)\\ \frac{d}{dx}\sigma(x) &= -1\left(\left(1+e^{-x}\right)^{(-2)}\right)\left(e^{-x}\right)(-1)\\ \frac{d}{dx}\sigma(x) &= \frac{(e^{-x})}{(1+e^{-x})^2}\end{aligned} \tag{11}$$

We can simplify the above, by adding and subtracting 1 in the numerator and rewrite it in terms of $\sigma(x)$.

$$\begin{aligned}\frac{d}{dx}\sigma(x) &= \frac{(1+e^{-x}-1)}{(1+e^{-x})^2}\\ \frac{d}{dx}\sigma(x) &= \frac{(1+e^{-x})}{(1+e^{-x})^2} - \frac{1}{(1+e^{-x})^2}\\ \frac{d}{dx}\sigma(x) &= \frac{1}{(1+e^{-x})}\left(1-\frac{1}{(1+e^{-x})}\right)\\ \therefore \frac{d}{dx}\sigma(x) &= \sigma(x)(1-\sigma(x))\end{aligned} \tag{12}$$

Q1.5

We have been given the following equation:

$$\begin{gathered} y = x^T W + b \\ y_j = x_1 W_{1j} + b_j + x_2 W_{2j} + b_j + \cdots + x_m W_{mj} + b_j \end{gathered} \tag{13}$$

We can represent the derivatives with respect to W as follows.

$$\begin{gathered} \frac{\partial J}{\partial W} = \frac{\partial J}{\partial y} \frac{\partial y}{\partial W} \\ where, \frac{\partial J}{\partial y} = \delta \in \mathbb{R}^{k \times 1} \\ \frac{\partial J}{\partial W} = \delta \frac{\partial y}{\partial W} \end{gathered} \tag{14}$$

Similarly, we can represent the derivatives with respect to x as follows.

$$\begin{gathered} \frac{\partial J}{\partial x} = \frac{\partial J}{\partial y} \frac{\partial y}{\partial x} \\ \frac{\partial J}{\partial x} = \delta \frac{\partial y}{\partial x} \end{gathered} \tag{15}$$

The last term above, $\frac{\partial y}{\partial W}$ and $\frac{\partial y}{\partial x}$ can be written in the matrix form as below.

$$\begin{gathered} \frac{\partial y}{\partial vec(W)} = \begin{bmatrix} \frac{\partial y_1}{\partial W_{11}} & \frac{\partial y_1}{\partial W_{21}} & \cdots & \frac{\partial y_1}{\partial W_{m1}} \\ \frac{\partial y_1}{\partial W_{12}} & \frac{\partial y_1}{\partial W_{22}} & \cdots & \frac{\partial y_1}{\partial W_{m2}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y_1}{\partial W_{1n}} & \frac{\partial y_1}{\partial W_{2n}} & \cdots & \frac{\partial y_1}{\partial W_{mn}} \end{bmatrix} \begin{bmatrix} \frac{\partial y_2}{\partial W_{11}} & \frac{\partial y_2}{\partial W_{21}} & \cdots & \frac{\partial y_2}{\partial W_{m1}} \\ \frac{\partial y_2}{\partial W_{12}} & \frac{\partial y_2}{\partial W_{22}} & \cdots & \frac{\partial y_2}{\partial W_{m2}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y_2}{\partial W_{1n}} & \frac{\partial y_2}{\partial W_{2n}} & \cdots & \frac{\partial y_2}{\partial W_{mn}} \end{bmatrix} \cdots \\ \frac{\partial y}{\partial x} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \cdots & \frac{\partial y_1}{\partial x_m} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} & \cdots & \frac{\partial y_2}{\partial x_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y_n}{\partial x_1} & \frac{\partial y_n}{\partial x_2} & \cdots & \frac{\partial y_n}{\partial x_m} \end{bmatrix} \end{gathered} \tag{16}$$

Let us take equation 13 and do the partial derivative for just the first elements with respect to W_{1j} and x_1. We get the following.

$$\begin{gathered} \frac{\partial y_1}{\partial W_{11}} = x_1 \\ \frac{\partial y_1}{\partial x_1} = W_{11} \end{gathered} \tag{17}$$

Now, doing the same for all the elements and putting them in the matrix form, we get the following.

$$\frac{\partial y}{\partial vec(W)} = \begin{bmatrix} x_1 & x_2 & \cdots & x_m \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & \cdots & 0 \\ x_1 & x_2 & \cdots & x_m \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} \cdots \tag{18}$$

On simplifying and rearranging this sparse matrix, we get the following.

$$\therefore \frac{\partial J}{\partial W} = x^T \delta \tag{19}$$

$$\frac{\partial y}{\partial x} = \begin{bmatrix} W_{11} & W_{21} & \cdots & W_{m1} \\ W_{12} & W_{22} & \cdots & W_{m2} \\ \vdots & \vdots & \ddots & \vdots \\ W_{1n} & W_{2n} & \cdots & W_{mn} \end{bmatrix} = W^T$$
$$\therefore \frac{\partial J}{\partial x} = \delta W^T \tag{20}$$

Q1.6

1.6.1

The range of the output of sigmoid function is [0,1] for any input value. While training the network using gradient descent, if large changes in the input lead to such small changes in the output, the gradient will be very small (for sigmoid, the interval of its derivative is [0, 0.25] as seen in figure 1). If sigmoid is used for successive layers and the layer outputs keep getting mapped to smaller and smaller regions successively, we end up having the vanishing gradient problem and the training might stall.

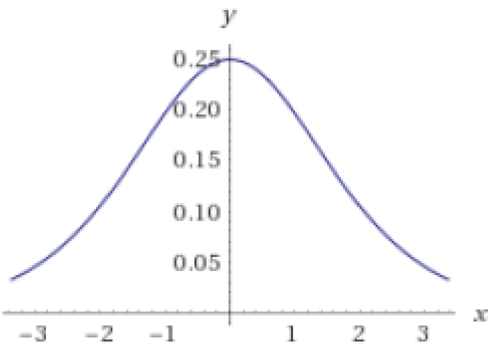


Figure 1: Plot of derivative of sigmoid

1.6.2

The tanh output range is interval [-1, 1] and sigmoid output range is interval [0, 1]. As per Yan LeCun's *Efficient BackProp* paper, convergence is usually faster if the average of each input variable over the training set is close to zero, which is true with tanh. The sigmoid activation function has the problem of saturating at 0 and 1, while tanh saturates at 1 and -1. So if the activity in the network during training is close to 0 then the gradient for the sigmoid activation function may go to 0. Therefore tanh is preferred.

1.6.3

The maximum value of derivative of tanh is up to 1 (As seen in figure 2) compared to 0.25 with sigmoid.Therefore, tanh leads to greater updates in the parameters during gradient descent and is better than sigmoid. Sigmoid has vanishing gradient problem if parameters are not updated properly in multiple layers.

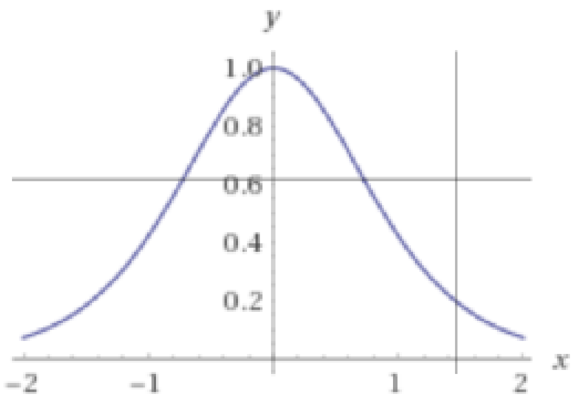


Figure 2: Plot of derivative of tanh

1.6.4

The derivation is as follows.

$$
\begin{aligned}
\sigma(x) &= \frac{e^x}{1+e^x} \\
\sigma(2x) &= \frac{e^{2x}}{1+e^{2x}} \\
e^{2x} &= \frac{\sigma(2x)}{1-\sigma(2x)} \\
\tanh(x) &= \frac{1-e^{-2x}}{1+e^{-2x}} \\
\tanh(x) &= \frac{1-\frac{1}{e^{2x}}}{1+\frac{1}{e^{2x}}} \\
\tanh(x) &= \frac{1-\frac{1-\sigma(2x)}{\sigma(2x)}}{1+\frac{1-\sigma(2x)}{\sigma(2x)}} \\
\tanh(x) &= \frac{\frac{\sigma(2x)-1+\sigma(2x)}{\sigma(2x)}}{\frac{\sigma(2x)+1-\sigma(2x)}{\sigma(2x)}} \\
\therefore \tanh(x) &= 2\sigma(2x)-1
\end{aligned}
\tag{21}
$$

Thus, it can be shown that tanh is a scaled and shifted version of the sigmoid.

2 Implement a Fully Connected Network

2.1

Q2.1.1

If we initialize a network weights with zeros in the back-propagation algorithm, the zero weights will be multiplied by delta, there will be no change and our training will have no effect on the weights.Also, if all the neurons are initiated with the same weight, they will follow the same gradient and could end up learning the same function.

Q2.1.3

We initialize with random numbers as it increases the entropy of the system and can increase the chances of finding the local minima faster.Also, if all the neurons are initiated with the same weights, they will follow the same gradient and could end up learning the same function. If the weights are not scaled properly, it could lead to vanishing or exploding gradient problems over multiple layers, which could result in slow training as the gradient descent can get stuck on flat or saturated regions of the activation function curves.

3 Training Models

Q3.1

The validation accuracy of 75% was obtained with **learning rate = 0.001 or 1e-3** and **batch size=32**. The plots of loss and accuracy of the model trained for 200 epochs are shown below.

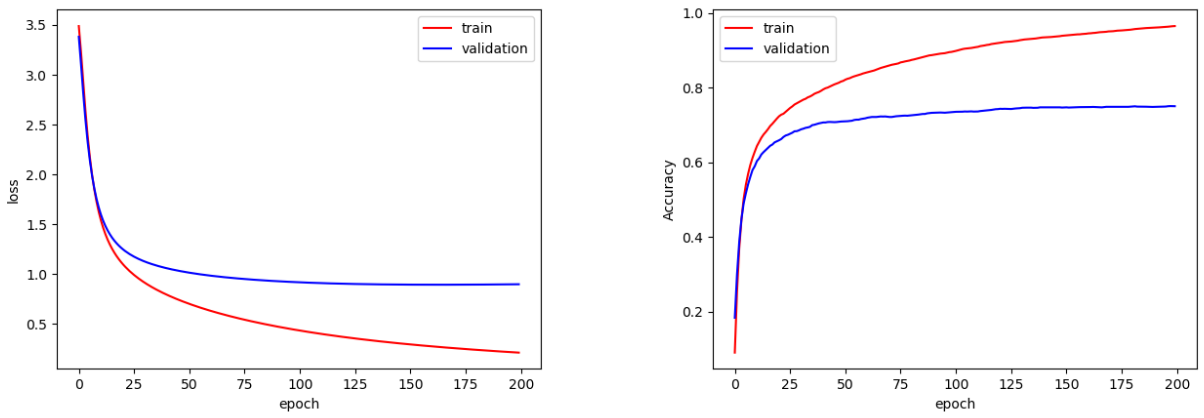


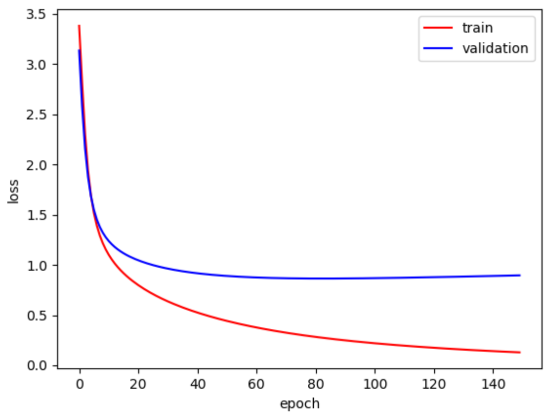
Figure 3: Loss and accuracy curves for learning rate = 1e-3

```
Train Accuracy: 0.96
Validation Accuracy: 0.75
```

Q3.2

1) Highest Accuracy

The best validation accuracy of 76.02% was obtained with **learning rate = 0.002 or 2e-3** and **batch size=32**. The plots of loss and accuracy of the model trained for 150 epochs are shown below.



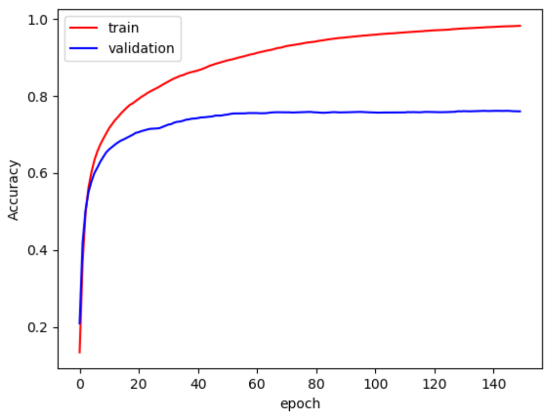


Figure 4: Loss and accuracy curves for learning rate = 2e-3

```
Train Accuracy: 0.98
Validation Accuracy: 0.76
```

2) Higher Learning Rate
When the learning rate was 0.02 (10 times higher), the losses were higher than the previous case while the accuracy was lower. The curves were also more bumpy because the step size was too large.

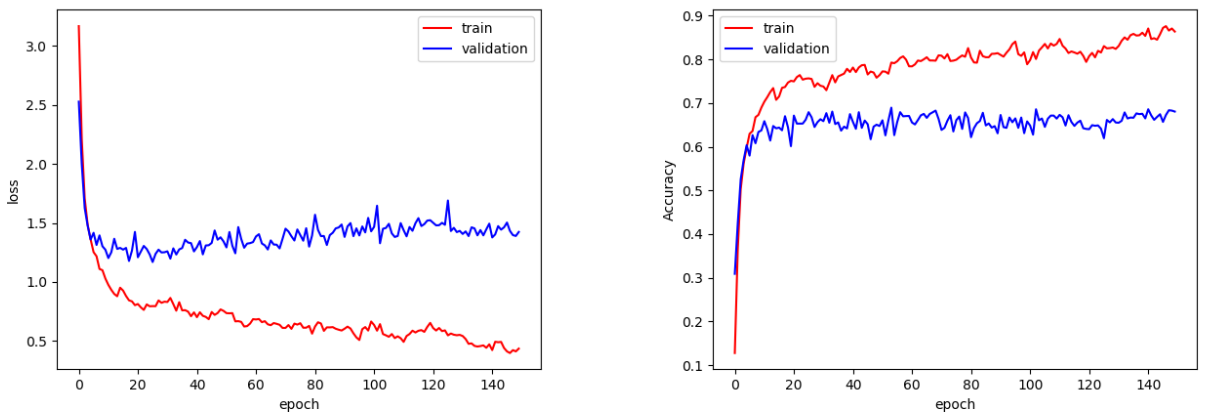


Figure 5: Loss and accuracy curves for learning rate = 0.02

```
Train Accuracy: 0.87
Validation Accuracy: 0.68
```

3) Lower Learning Rate
When the learning rate was 0.0002 (1/10 times), the curves are smooth as those shown, but because the step size was too small, it did not converge to the optimum within the maximum number of iterations (150) that we set.

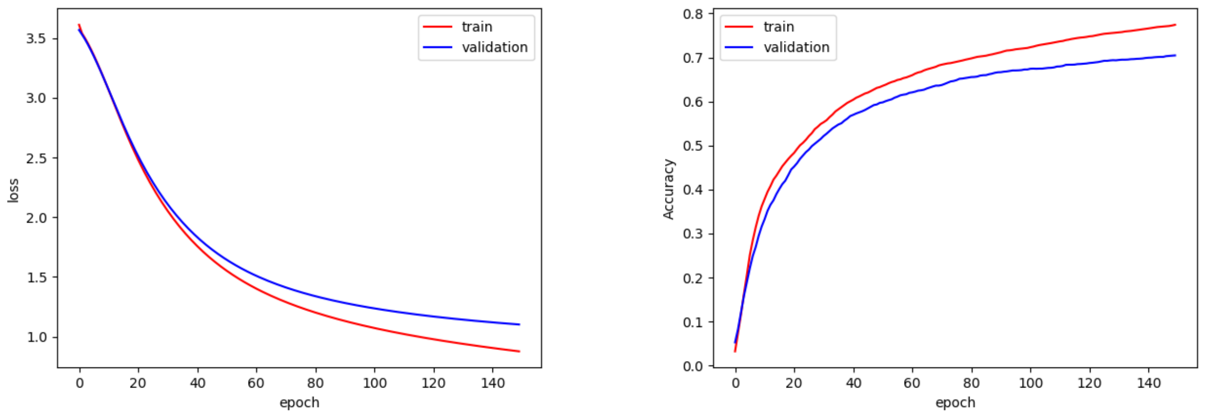


Figure 6: Loss and accuracy curves for learning rate = 2e-4

```
Train Accuracy: 0.77
Validation Accuracy: 0.70
```

Q3.3

The plots of first layer weights and network weights of the best learning rate (2e-3) that gave the highest validation accuracy are given below.

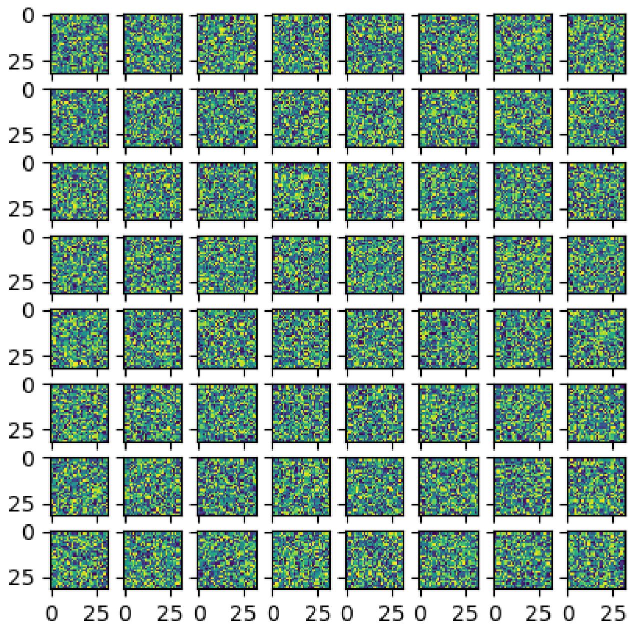


Figure 7: First layer weights immediately after initialization

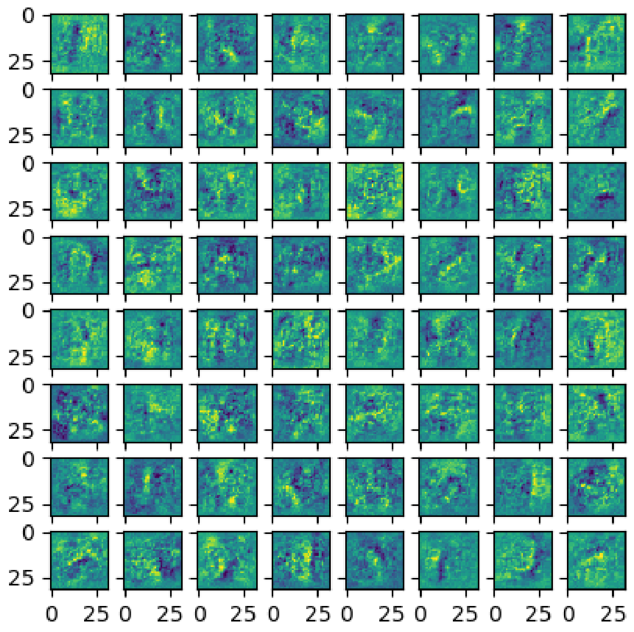


Figure 8: First layer weights after training for 150 epochs with learning rate 2e-3

On comparing the two figures, we can see that the initial weights have no patterns in them as we initialized the layers with random uniform distribution. The weights after 150 epoch training have some more clear patterns, visible strokes of the letters and numbers which look somewhat averaged over the dataset

Q3.4

Below is the visualization of the confusion matrix for the best model, with learning rate = 2e-3.

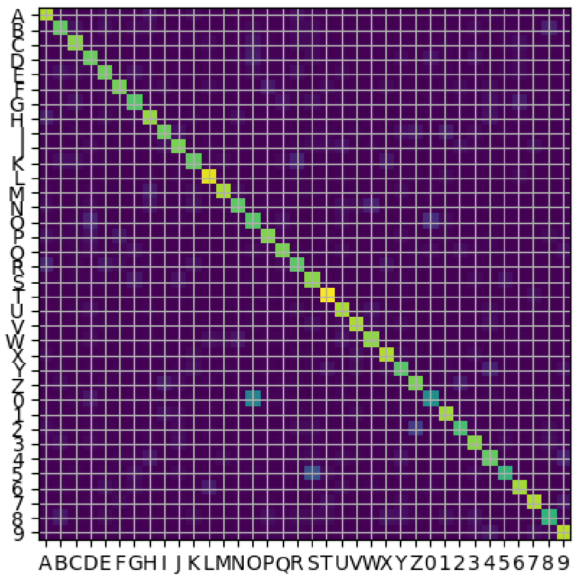


Figure 9: Confusion matrix on the NIST36 test dataset

From the above confusion matrix it can be seen that the brighter the grid, more number of correctly predicted labels in the grid. Since the main diagonal is the heaviest, we could conclude that the results are pretty good.
After training 150 epochs the top few pairs that are commonly confused are number ’0’ and letter ’O’, ’2’ and ’Z’, ’5’ and ’S’, ’4’ and ’Y’, ’0’ and ’D’, which are often misjudged manually too.

4 Extract Text From Images

4.1

The assumptions the sample method makes are as follows:

1. The stroke width, style of writing of the characters extracted are assumed to be similar to the NIST36 dataset that were trained. Also, the pixel values of the extracted characters are assumed to be similar to the ones in the dataset, however the dataset does not have perfect binary black-and-white images upon inspection.

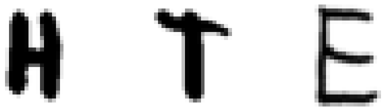
   

   Figure 10: Character detection failure due to style of writing

   From the above image, the stroke widths are different than in the dataset and these images might fail to be detected correctly

2. The cropped image are assumed to not have any noises, any overlap between two letters and parts of letters to be fully connected.

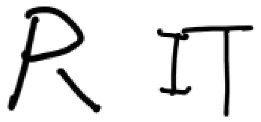
   

   Figure 11: Character Detection failure due to incompleteness of letters

   In the above image, the first letter has incomplete parts, and second letter and third letters are overlapping. This image might fail to be detected correctly

Q4.3

The output of the findLetters() are given below. All the letters were found and circled by the red rectangles.

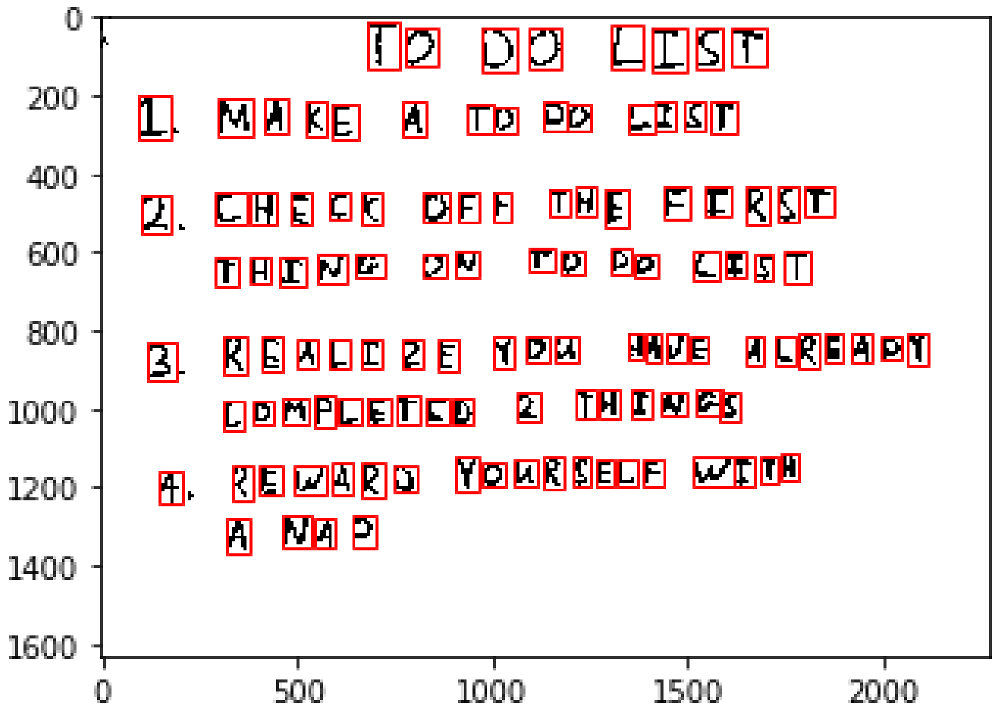


Figure 12: Text extraction results on 01_list.jpg

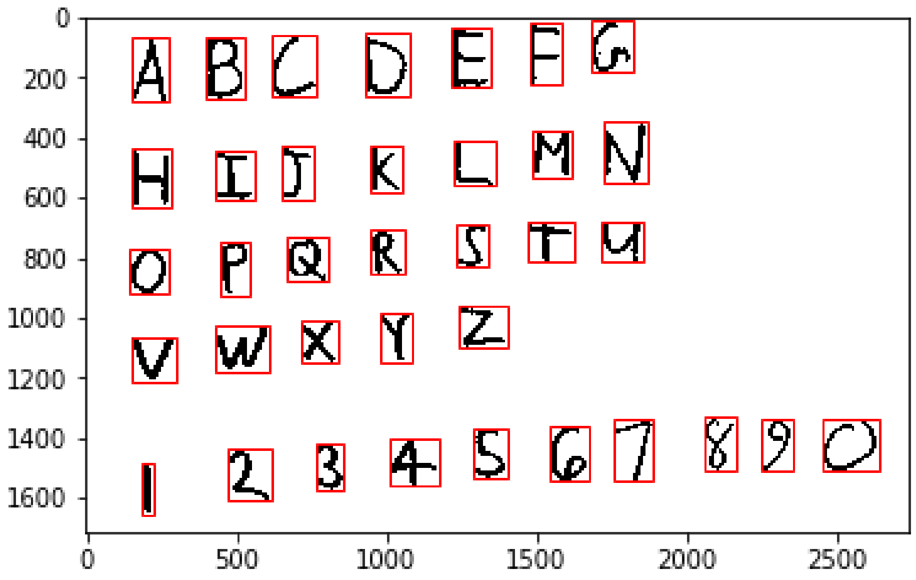


Figure 13: Text extraction results on 02_letters.jpg

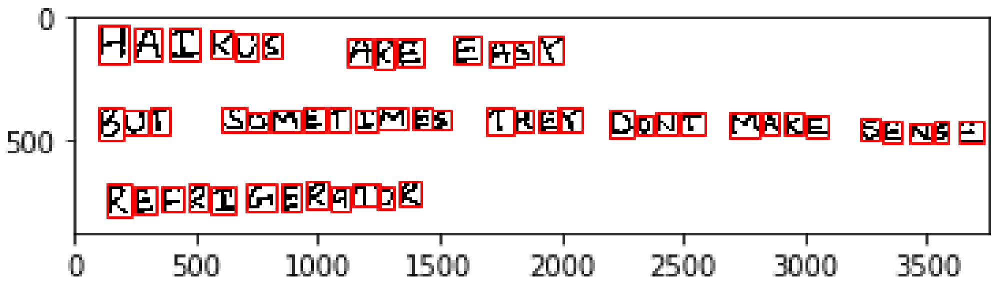


Figure 14: Text extraction results on 03_haiku.jpg

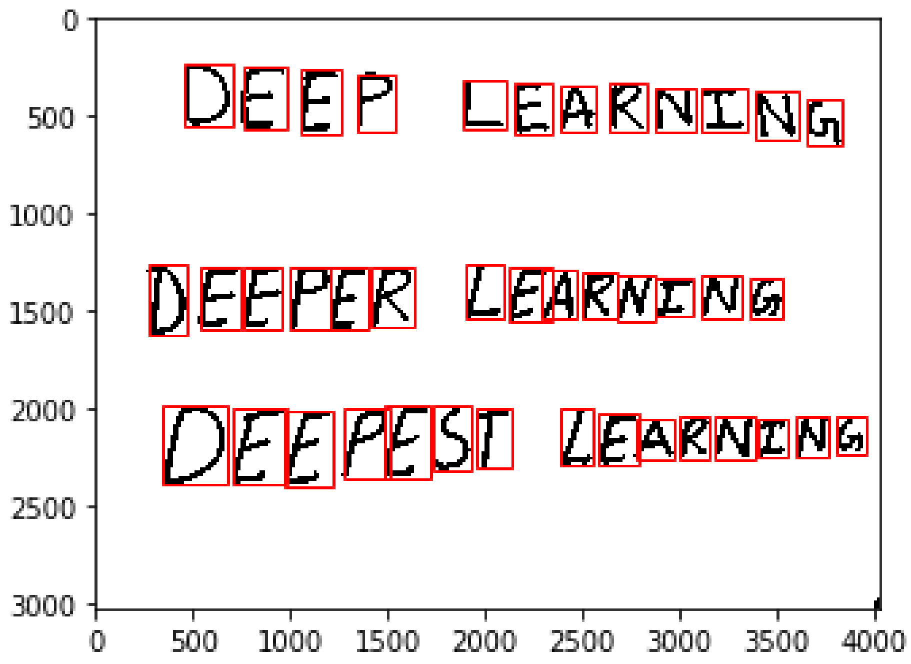


Figure 15: Text extraction results on 04_deep.jpg

From the above images it can be seen that the algorithm was able to detect all the letters in the given images with 100% accuracy.

Q4.4

In order to make the detected images look like the images from the training set, cropping and padding was done which is shown below:

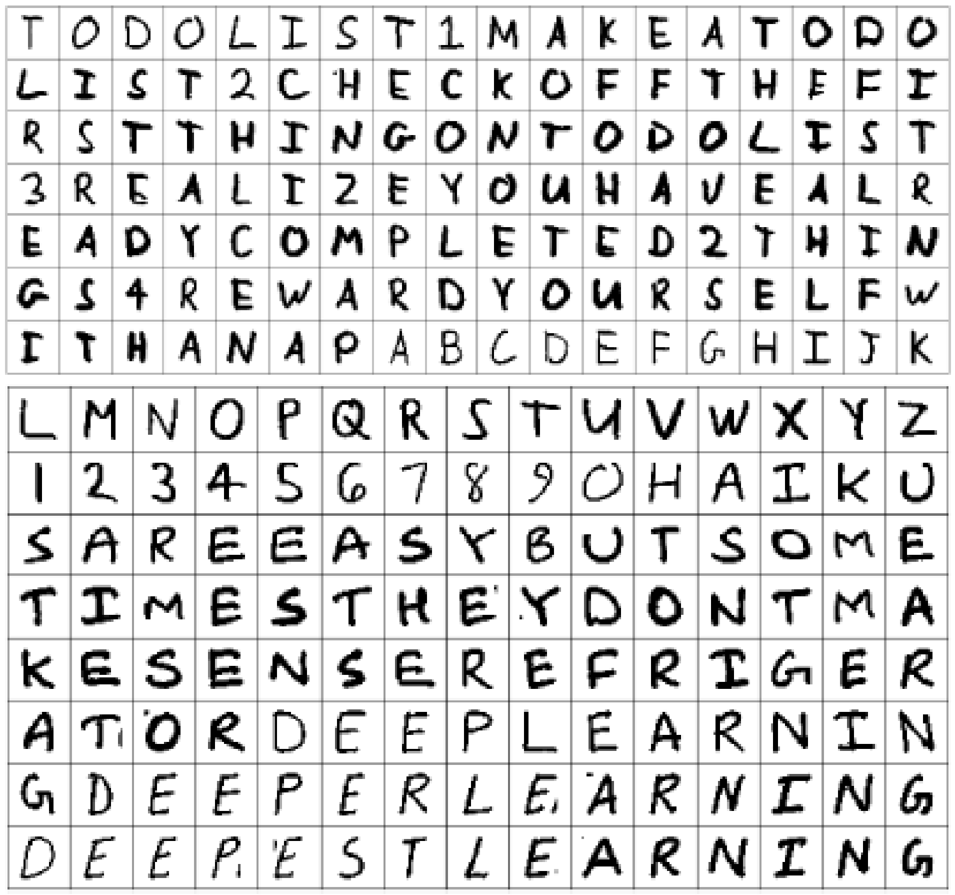


Figure 16: Text extraction results after cropping and padding

The extracted text and its accuracy from the detection is as below:

```
TODOLIGT
IHAXEATO2OLI5T
2CH2EK0F87HEFIRST
THINGONTOOOLIIT
3R8ALIZEYOUHAVEALR2ADY
COMPL8T2DZIHINGI
4R8WARDYOURSELPW8TR
ANAP

Accuracy: 71.30%
```

Figure 17: Text detection results and accuracy for 01_list.jpg

```
2BCDEPG
HIIKLMN
O9QRSTU
VWXYZ
1Z34S67I7D

Accuracy: 72.22%
```

Figure 18: Text detection results and accuracy for 02_letters.jpg

```
HAIKUGAREEASX
BWTSQMETIMESTREXDONTMAK2SEMGE
REFRIGERATOR

Accuracy: 79.63%
```

Figure 19: Text detection results and accuracy for 03_haiku.jpg

```
DEEPLEARMING
DEEPERLEARKING
DEERESTLEARNIHG

Accuracy: 90.24%
```

Figure 20: Text detection results and accuracy for 04_deep.jpg

We can see that, overall, the results were pretty good, ”04_deep.jpg” has the highest detection. accuracy as the texts were similar to that of training set.

5 Image Compression with Autoencoders

5.2 Training the Autoencoder

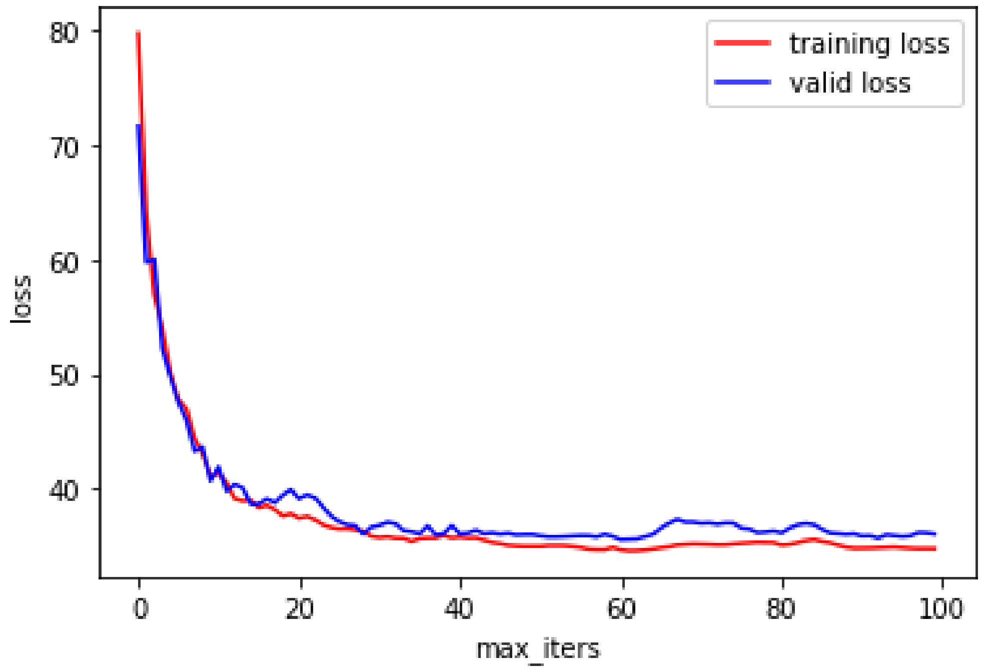


Figure 21: Training and validation loss curve for the Autoencoder

It can be seen that the training loss decreases as the training progresses with a initial steep drop and a very flat drop in loss towards the end. Also, it can be observed that the validation loss is greater than the training loss.

5.3.1 Evaluating the Autoencoder

Below given are the 5 validation set classes and their reconstruction:

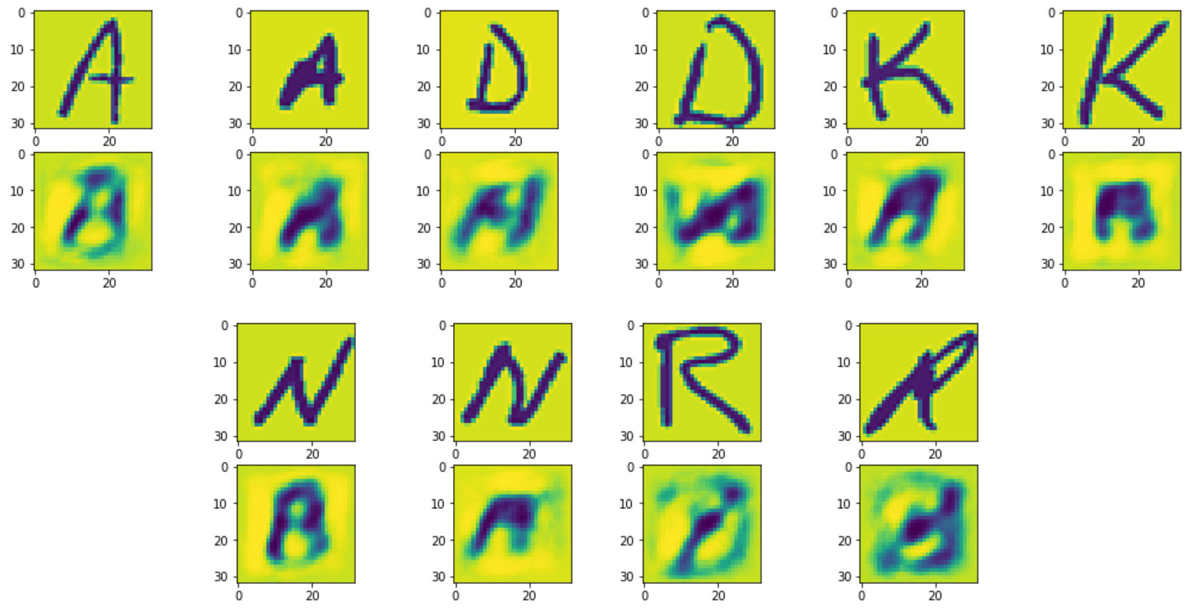


Figure 22: Reconstruction results of the Autoencoder for five classes from the NIST36 test dataset

The reconstruction results look more blurred and more interpolations existed between the edges of the letters and the background as compared to the orginal images. This is because we are reducing the dimensionality at the bottleneck of the Autoencoder.

5.3.2

The average Peak Signal-Noise Ratio (PSNR) across all the validation images was **14.913**.

6 Pytorch

6.1

6.1.1

The plots of loss and accuracy of re-written and re-trained fully-connected network with NIST36 Dataset is shown below:

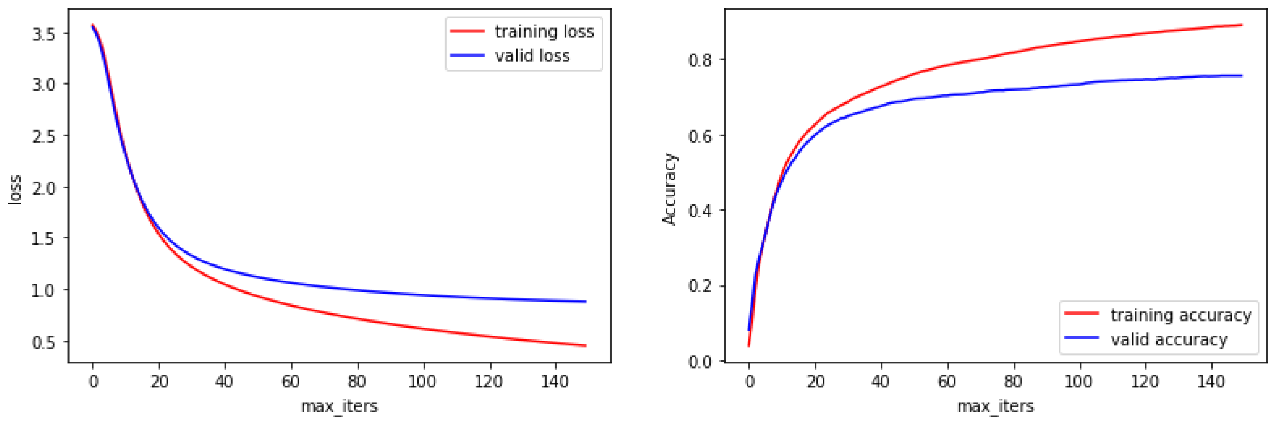


Figure 23: Loss and accuracy curves for Fully Connected Network with NIST36 Dataset

6.1.2

Results of convolution Neural Network (LeNet-5) with NIST36 Dataset is shown below:

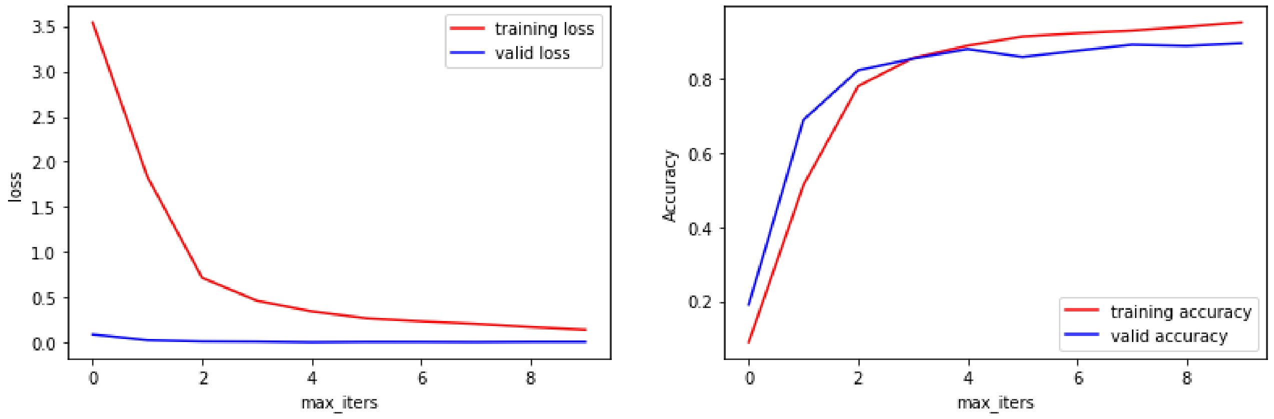


Figure 24: Loss and accuracy curves for Convolution Neural Network (LeNet-5) with NIST36 Dataset

It can be seen that the Convolution Neural Network model has higher training and validation accuracy than the Fully connected network model for the same dataset.
The number of iterations required by the Convolution Neural Network model to reach the higher accuracy is also less as compared to the Fully connected network model.

6.1.3

Results of convolution Neural Network with CIFAR-10 is shown below:

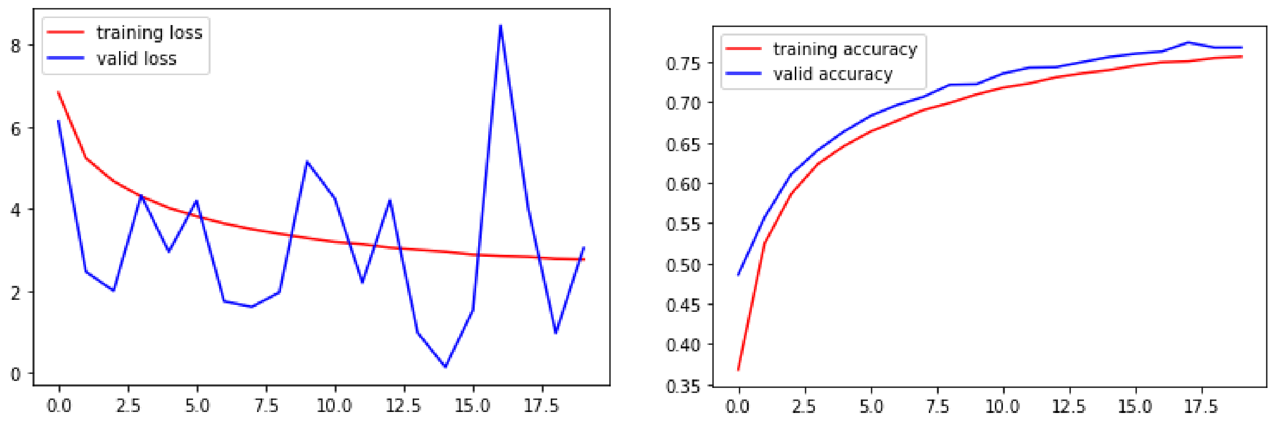


Figure 25: Loss and accuracy curves for Convolution Neural Network with CIFAR-10 Dataset