Kartik Narula, knarula
HW5

1

1.1

We have,

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

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

Using c=max(x_i), makes all the numerator values non positive. It prevents overflow by making very large positive values smaller.

Kartik Narula, knarula
HW5

1.2

a) Sum of all elements in softmax(x) =1 and each value is in the range (0,1]

b) Probability distribution.

c) S_i maps each value to a exponential function, returning a positive value. The second step takes the sum of all softmax values- this is a normalization step. The third step divides each softmax value with the sum, returning a value between 0 and 1 which can be used as a probability.

Kartik Narula, knarula
HW5

1.3

Suppose a neural network with 1 hidden layer and 3 hidden units, which maps 3 inputs X_1, X_2 and X_3 to a single output Y.

Let Z_1, Z_2 and Z_3 denote the units in the hidden layer and W_{IJ} represents the weight, and b_i represents the bias.

We have,

$Z_1= W_{11}X_1+W_{12}X_2+W_{13}X_3+b_1$

$Z_2= W_{21}X_1+W_{22}X_2+W_{23}X_3+b_2$

$Z_3= W_{31}X_1+W_{32}X_2+W_{33}X_3+b_3$

And, $Y= W_1Z_1+ W_2Z_2+ W_3Z_3$

Or, $Y= W_1(W_{11}X_1+W_{12}X_2+W_{13}X_3+b_1) + W_2(W_{21}X_1+W_{22}X_2+W_{23}X_3+b_2)+ W_3(W_{31}X_1+W_{32}X_2+W_{33}X_3+b_3)$

Which can be written as,

$Y=AX_1+BX_2+CX_3+D$

Where $A= W_1W_{11}+ W_2W_{21}+ W_3W_{31}$

$B= W_1W_{12}+ W_2W_{22}+ W_3W_{32}$

$C= W_1W_{13}+ W_2W_{23}+ W_3W_{33}$

And $D= W_1b_1+ W_2b_2+ W_3b_3$

Hence, multi-layer neural network with no non linear activation is equivalent to a linear regression.

Kartik Narula, knarula
HW5

1.4

We have,

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

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

$$= \frac{0 - 1 \times e^{-x} \times (-1)}{(1+e^{-x})^2}$$

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

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

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

$$= \sigma(x)(1 - \sigma(x))$$

Kartik Narula, knarula
HW5

1.5

$y = WX^T + b$

which can be written in expanded matrix form as follows:

$$\begin{matrix} y_1 \\ y_2 \\ . \\ : \\ . \\ y_k \end{matrix} = \begin{matrix} W_{11} & W_{12} & . & . & W_{1d} \\ W_{21} & W_{22} & . & . & W_{2d} \\ . & . & . & . & . \\ . & . & . & . & . \\ W_{kd} & . & . & . & W_{kd} \end{matrix} \times \begin{matrix} x_1 \\ x_2 \\ . \\ , \\ x_d \end{matrix} + \begin{matrix} b_1 \\ b_2 \\ \vdots \\ \ddots \\ b_k \end{matrix}$$

We have,

$y_1 = W_{11}x_{1+} W_{12}x_2 + W_{13}x_3 + \ldots\ldots W_{1d}x_d$

$y_2 = W_{21}x_{1+} W_{22}x_2 + W_{23}x_3 + \ldots\ldots W_{2d}x_d$

.

$y_k = W_{k1}x_{1+} W_{k2}x_2 + W_{k3}x_3 + \ldots\ldots W_{kd}x_d$Type equation here.

If J represents the loss, let

$$\frac{\delta J}{\delta y} = \delta$$

Then, $\delta = \begin{matrix} \frac{\delta J}{\delta y_1} \\ \frac{\delta J}{\delta y_2} \\ \vdots \\ \frac{\delta J}{\delta y_k} \end{matrix} = \begin{matrix} \delta_1 \\ \delta_2 \\ \vdots \\ \delta_k \end{matrix}$

$$\frac{\delta J}{\delta W_{11}} = \frac{\delta J}{\delta y_1} \times \frac{\delta y_1}{\delta w_{11}} = \delta_1 \times x_1$$

Similarly, we have,

$$\frac{\delta J}{\delta W_{ij}} = \delta_i \times x_j$$

Writing in matrix form we get,

$$\boldsymbol{\frac{\delta J}{\delta W} = \delta X}$$

Kartik Narula, knarula
HW5

$\frac{\delta J}{\delta x_1} = \delta_1 W_{11} + \delta_2 W_{21} \ldots . + \delta_k W_{k1}$

Similarly we have,

$\frac{\delta J}{\delta x_i} = \delta_1 W_{1i} + \delta_2 W_{2i} \ldots . + \delta_k W_{ki}$

This can be written in matrix form as:

$\boldsymbol{\frac{\delta J}{\delta X} = \delta^T W}$

$\frac{\delta J}{\delta b_1} = \delta_1$

Similarly,

$\frac{\delta J}{\delta b_i} = \delta_i$

In matrix form we have,

$\boldsymbol{\frac{\delta J}{\delta b} = \delta}$

Kartik Narula, knarula
HW5

1.6

a.

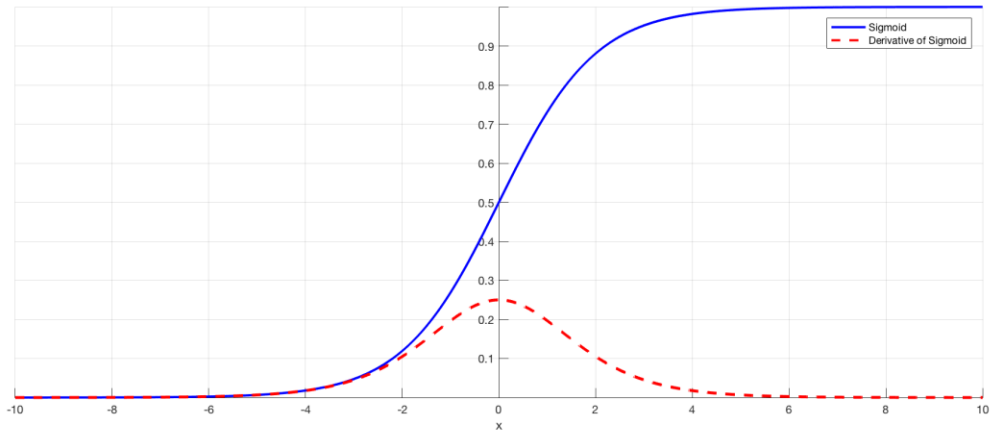


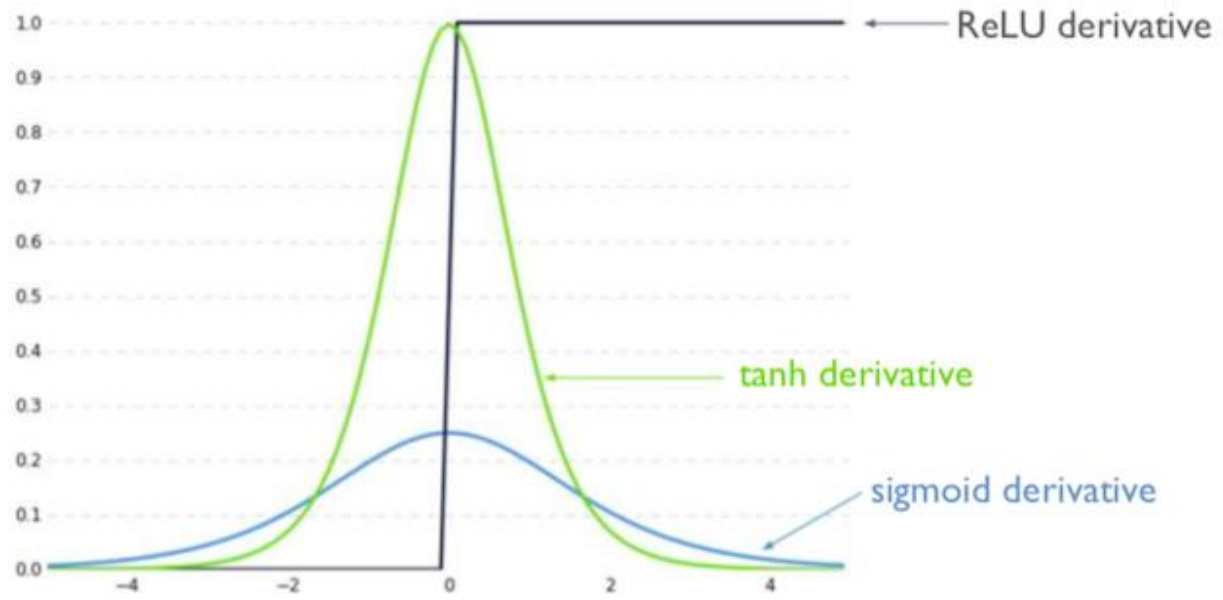
Figure 1: Sigmoid Function and it's derivative

The sigmoid activation function squishes the inputs to between 0 and 1. Looking at the above figure, the derivative becomes very small when the values are close to 0 or 1. During backpropogation, gradients are multiplied towards the end of the layer. It is not a problem for shallow networks, but for deeper networks the gradient becomes exceedingly small as we move towards the initial layers. Thus, the initial layers do not get updated at all.

b. Tanh ranges from (-1, 1) and sigmoid ranges from (0,1). Tanh is centered around the origin and outputs normalized values, which help convergence faster. Sigmoid maps all the values between 0 and 1, so a large negative value will be mapped to zero, while tanh will map it to a negative value. Hence, tanh will achieve faster convergence. Also, tanh has larger derivative values, so it doesn't have a vanishing gradient problem as the sigmoid.

Kartik Narula, knarula
HW5

c.



As we can see in the above image tanh derivative values are larger than the sigmoid. Hence it has less of the vanishing gradient problem.

d. We have,

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

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

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

Putting value of e^{-2x} in (1) we get,

tanh(x) = $\frac{1-\frac{1}{\sigma(2x)}-1}{1+\frac{1}{\sigma(2x)}-1}$ = $2\sigma(2x) - 1$

Hence, tanh(x) is a scaled and shifted version of sigmoid.

Kartik Narula, knarula
HW5

2

2.1.1

It is not a good idea to initialize the network with all zeros, because it will lead to all the neurons performing the same calculation in the training process and thus all the neurons will learn the same features. This is known as ‘not being able to break the symmetry’. We will get the same issue on initializing all the weights with the same constant value. Also initialization the weights with very small values would lead to vanishing gradients.

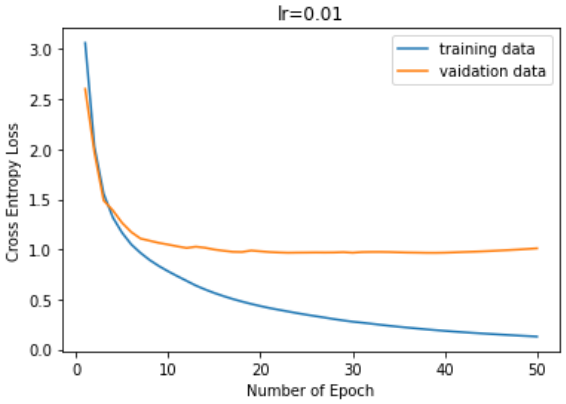
Kartik Narula, knarula
HW5

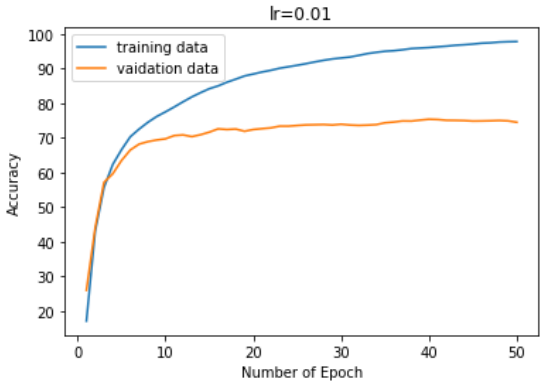
2.1.3

We initialize the weights with random values to break the symmetry. This makes it possible for different neurons to learn different features. Scaling each initialization with respect to the layer size helps to account for the variance of the backpropogated layer and prevents the vanishing gradient problem.

Kartik Narula, knarula
HW5

3.1





The learning rate is 0.01. Training set accuracy is 98.11 % and the validation set accuracy is 74.75%.

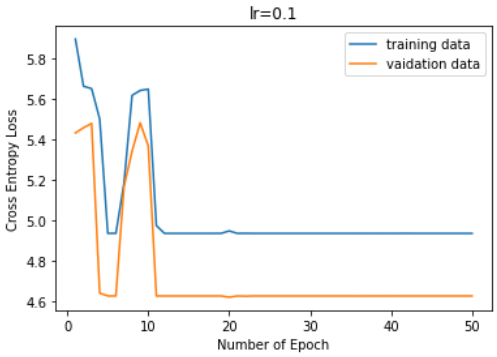
Kartik Narula, knarula
HW5

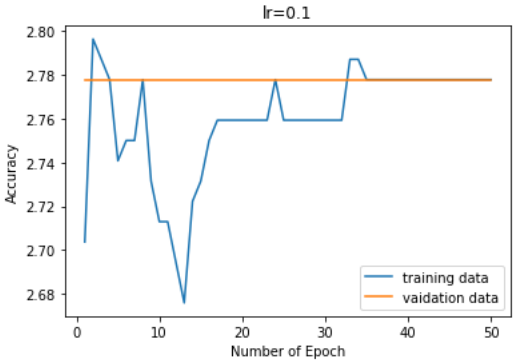
3.2

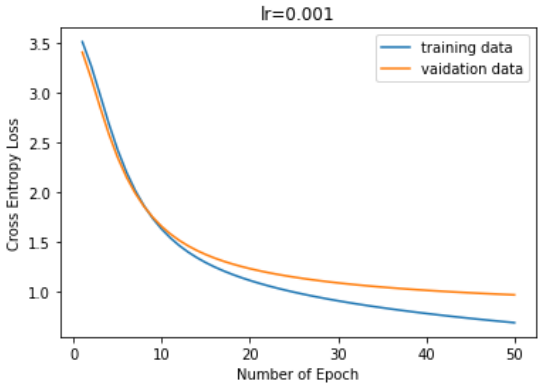
Learning rate determines how fast the weights are updated. When the learning rate is high, the weights update themselves quickly and oscillate over the iterations as shown. It also increases the chances of getting stuck at local minima.

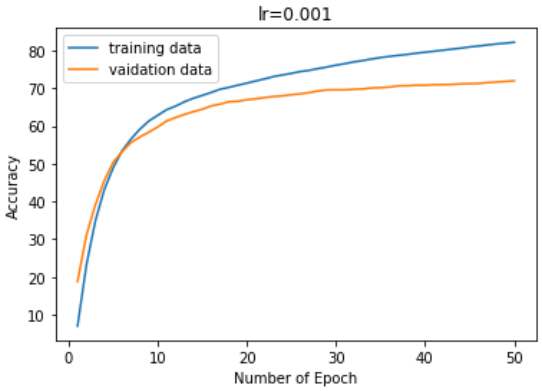
When the learning rate is small, optimum is reached albeit more slowly and greater number of iterations are required.

Best performance: learning rate – 0.01 and 50 epochs-validation accuracy :75.73%



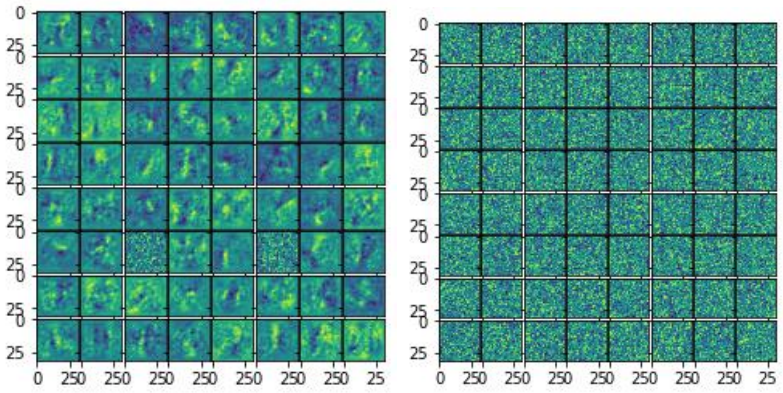






Kartik Narula, knarula
HW5

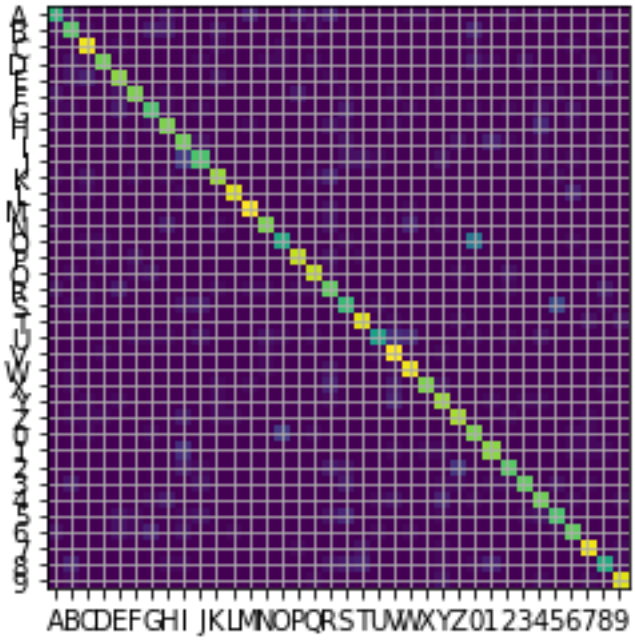
3.3



The image on the left show the visualization for the learned weights while the image on the right show the visualization for the initialized weights. The left image resembles some kind of learned features while the right image looks random(weights were initialized randomly).

Kartik Narula, knarula
HW5

3.4



Confusion matrix for the test set.

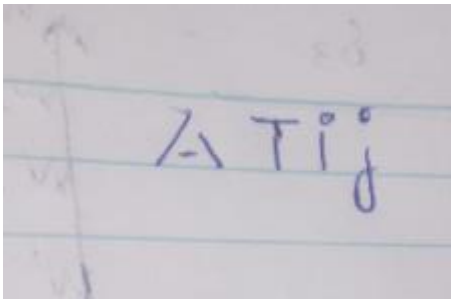
The most confused classes are: ‘o’ and ‘0’, ‘5’ and ‘S’, ‘2’ and ‘Z’ and ‘1’ and ‘I’. These characters have similar features and hence are often misidentified by the neural network.

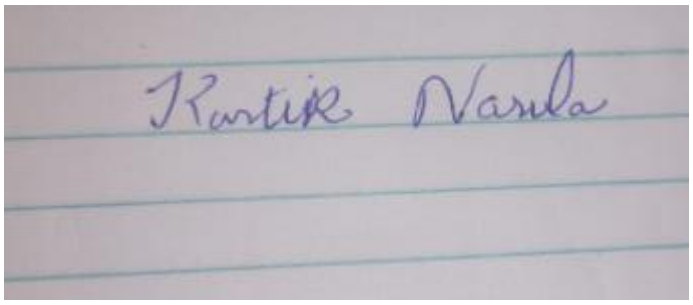
Kartik Narula, knarula
HW5

4.1 Two assumptions made by this method are:

i. Letters have spaces between them and can be separated by bounding boxes. This may not hold true for handwritten texts.

ii. The algorithm sometimes detects multiple bounding boxes for the same letter- These include letters like T , i and j. It fails when the letters are not fully connected.

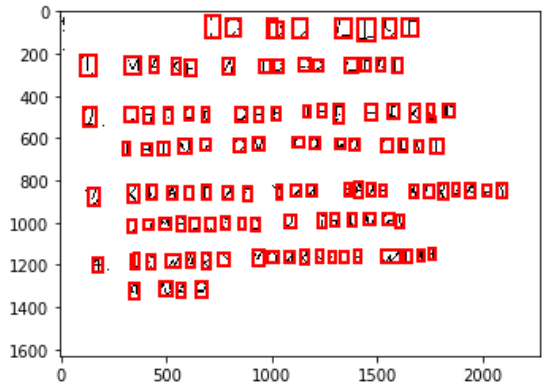


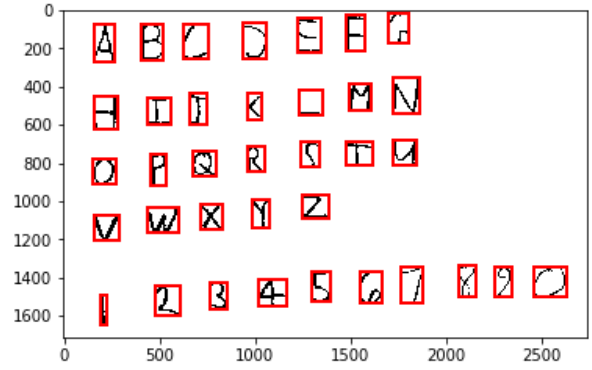


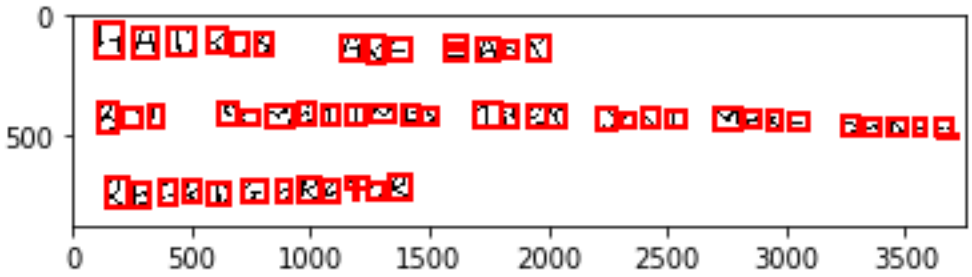
Possible failure examples

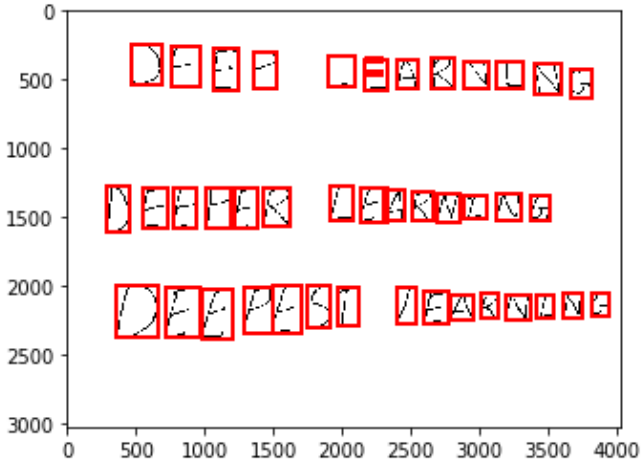
Kartik Narula, knarula
HW5

4.3



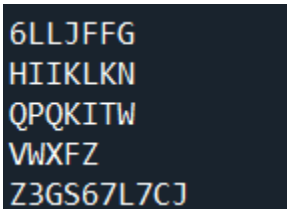






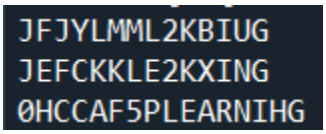
Kartik Narula, knarula
HW5

4.4

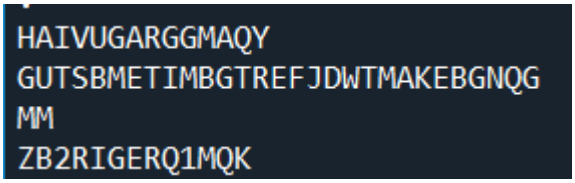


02_letters

01_list



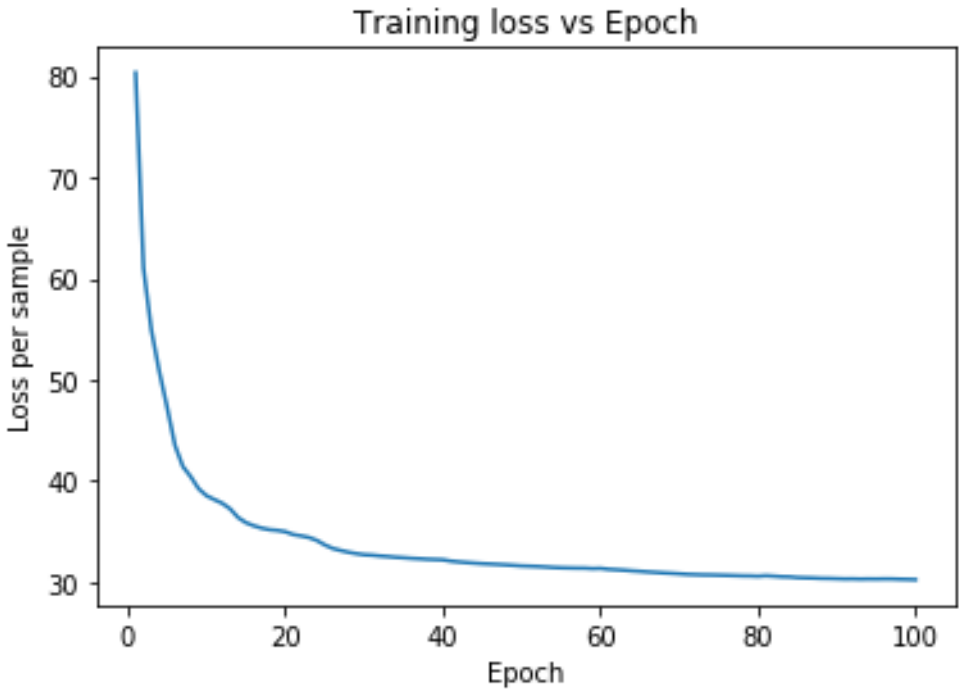
04_Deep



03_haiku

Kartik Narula, knarula
HW5

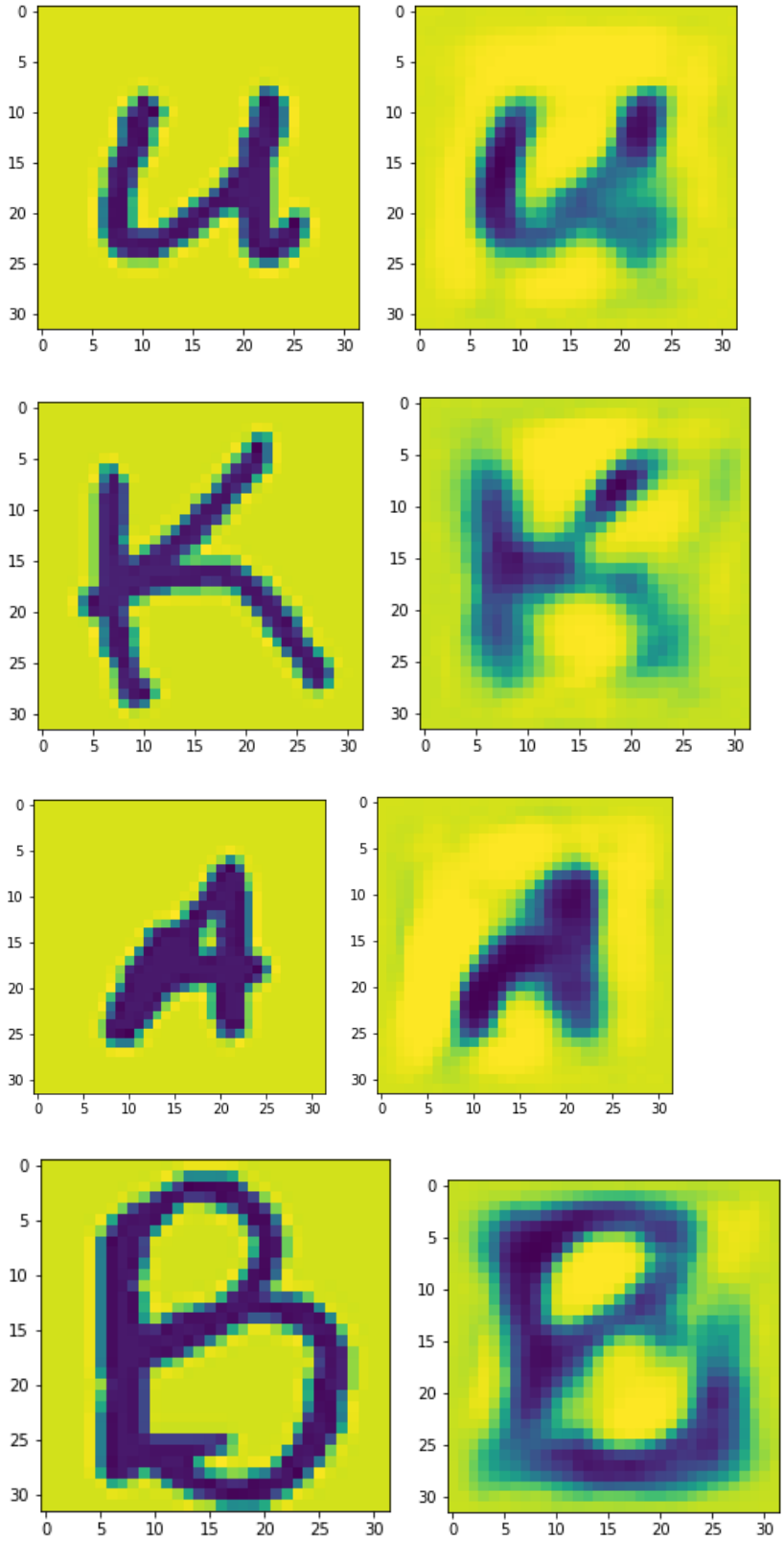
5.2



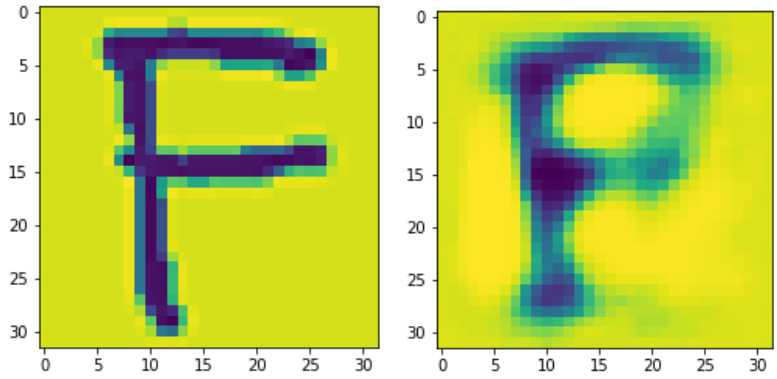
The loss increases sharply till about 20 iterations and the decline is much smoother after that. We should stop training at about 40-50 iterations to avoid overfitting.

Kartik Narula, knarula
HW5

5.3.1



Kartik Narula, knarula
HW5



The images outputted by the autoencoder are blurred and captured the essential features but some information(such as edges) has been lost.

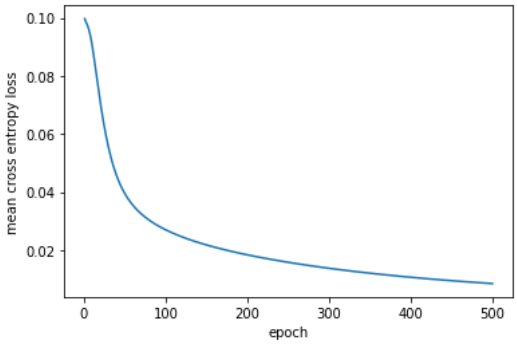
Kartik Narula, knarula
HW5

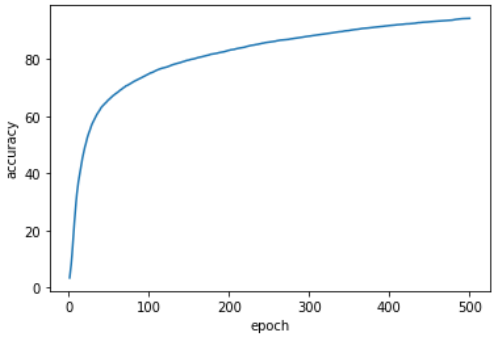
5.3.2

The average PSNR value reported for the validation images is 15.07

Kartik Narula, knarula
HW5

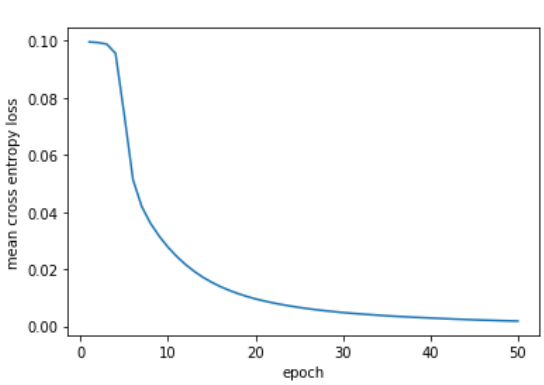
6.1.1

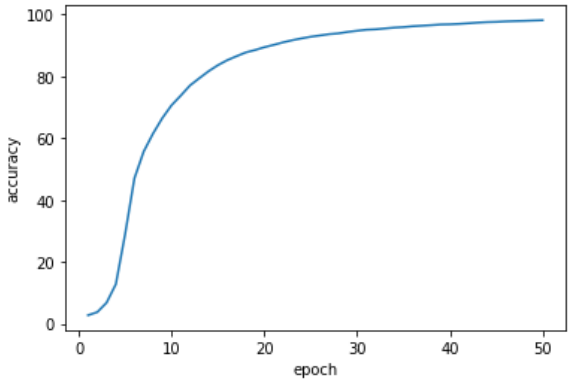




Kartik Narula, knarula
HW5

6.1.2

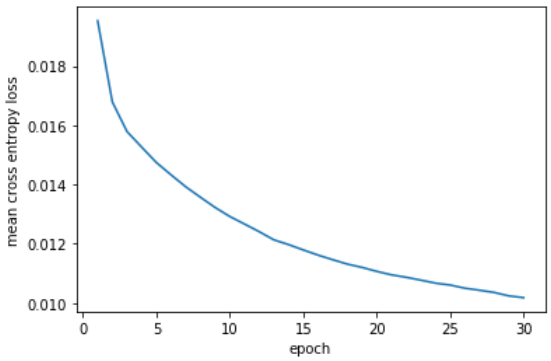


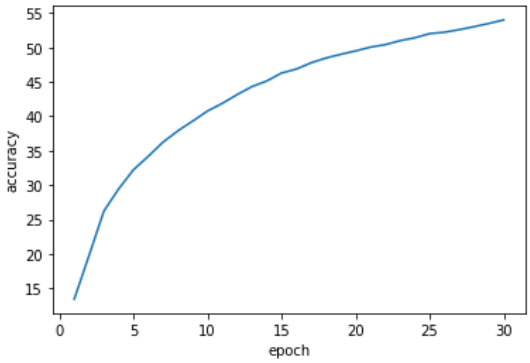


The network takes fewer epochs to converge while using a convolution neural network.

Kartik Narula, knarula
HW5

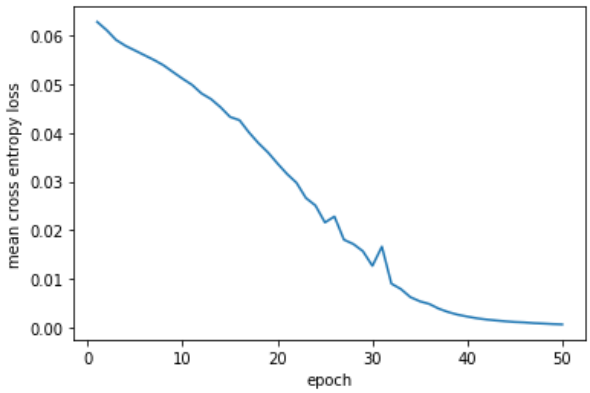
6.1.3

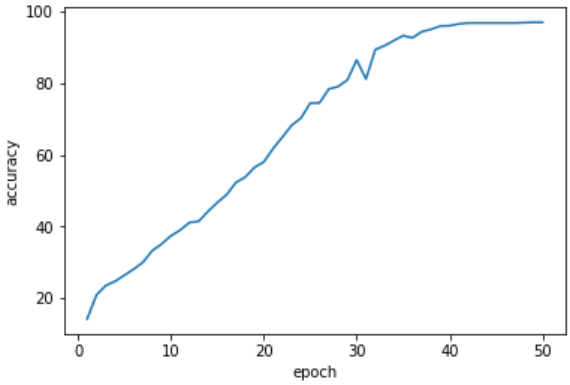




Kartik Narula, knarula
HW5

6.1.4





Test accuracy was observed to be about 45 percent which is less than observed in HW1. Improvements in the network architecture will certainly help improve the test accuracy.