16-720 B Assignment5

Sajal Maheshwari
sajalm@andrew.cmu.edu

November 23, 2019

1 Theory

1.1

$$softmax(x+c) = \frac{e^{x_i+c}}{\sum e^{x_i+c}} \tag{1}$$

$$softmax(x+c) = \frac{e^{x_i}e^{c}}{e^{c}\sum e^{x_i}} \tag{2}$$

$$softmax(x+c) = \frac{e^{x_i}}{\sum e^{x_i}} = softmax(x) \tag{3}$$

We should use the value of c as $\max(-x_i)$ because that will ensure that the coefficient of the exponential term is always greater than 0. This will in turn ensure that the numerator is always greater than 1. Since that is the case, our final softmax terms will contain all values in the numerator and the summation part of the denominator as grater than 1, which will help in avoiding divisions between extremely small numbers which usually would throw floating point errors/make the function unstable.

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 which represents the probability of occurrence of one of each of the elements of the vector.
The first step makes each dimension of the vector positive. Step 2 sums up each of the dimension while the third step actually makes it a probability distribution.

1.3

We first show an example with two linear layers. Let $y_1 = W_2^T(W_1^T X + b_1) + b_2$. On expanding the expression, we get $y_1 = W_2 W_1^T X + W_2 b_1) + b_2$. This is a linear layer with weights equal to $W_2 W_1$ and bias equal to $W_2 b_1 + b_2$. This same logic can be expanded inductively to multiple layers.

(4)

1.4

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

1.5

The expression for each element of the output vector is : $y_j = \sum_{i=1}^{d} x_i W_{ij} + b_j$

Differentiating y_j w.r.t. W_{ij}, x_i and b_j, we get the following expressions : $\frac{\partial y_j}{\partial W_{ij}} = x_i, \quad \frac{\partial y_j}{\partial x_i} = W_{ij}, \quad \frac{\partial y_j}{\partial b_j} = 1$

Using the chain rule, we can then write the expression of loss w.r.t. $\mathbf{x}$, $\mathbf{W}$ and $\mathbf{b}$ as :

$$\frac{\partial J}{\partial W} = \sum_{j=1}^{k} \frac{\partial J}{\partial y_j} \frac{\partial y_j}{\partial W} = x\delta^T \in R^{d \times k}$$

$$\frac{\partial J}{\partial x} = \sum_{j=1}^{k} \frac{\partial J}{\partial y_j} \frac{\partial y_j}{\partial x} = W\delta \in R^{d \times 1}$$

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

1.6

The sigmoid activation funtion does rapidly reaches very close to its extremum values due to which the gradient values are very less for an iteration, leading to the problem of vanishing gradients.
The tanh activation function has the range (-1, 1). We prefer tanh because it does not have the problem of the vanishing gradient as much as the sigmoid function.
The tanh does not have the problem of the vanishing gradient as severe as the sigmoid function because as the extreme negative values also get mapped to the positive values unlike the sigmoid function.

$$tanh(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{5}$$

$$tanh(x) = \frac{1}{1 + e^{-2x}} - \frac{e^{-2x}}{1 + e^{-2x}} \tag{6}$$

$$tanh(x) = \frac{1}{1 + e^{-2x}} - \frac{1 + e^{-2x}}{1 + e^{-2x}} + \frac{1}{1 + e^{-2x}} \tag{7}$$

$$tanh(x) = 2 * \sigma(2x) - 1 \tag{8}$$

2

2.1

2.1.1

We do not initialize a network with all zeros because in that case the gradient of all the weights will be pointing in a single direction for every iteration, significantly increasing the convergence time of the network. A zero-initialized network after training can contain the same set of weights corresponding to all the output nodes, i.e. for a given input the output nodes show equal values. This will be essentially equal to a uniform random probability of the output class being detected.

2.1.2

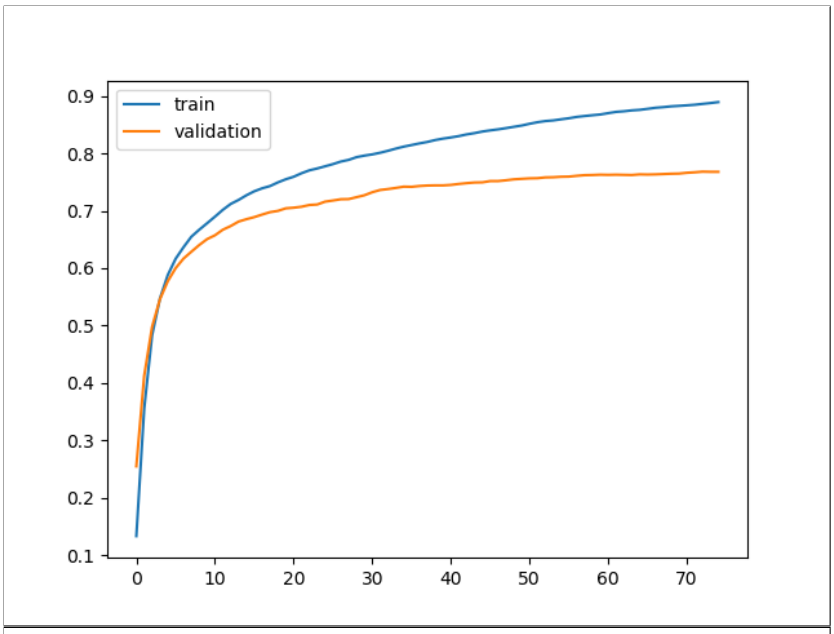
2.1.3

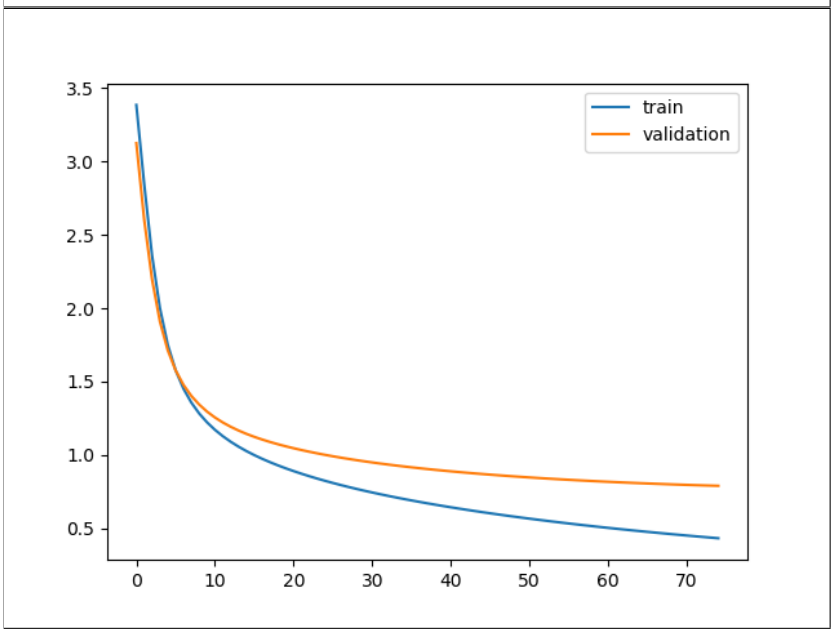
We initialize the weights with random values to break the symmetry we get due to the zero initializations. We scale the initialization based on the network layers because if the weights are close to zero values in the deeper layers, during backpropogation, the gradients will tend to be very low at the layers close to the input. Therefore, the deeper weights should have higher initial values to begin with, which is achieved by scaling the initialization based on the network layers.

3 Metric reconstruction

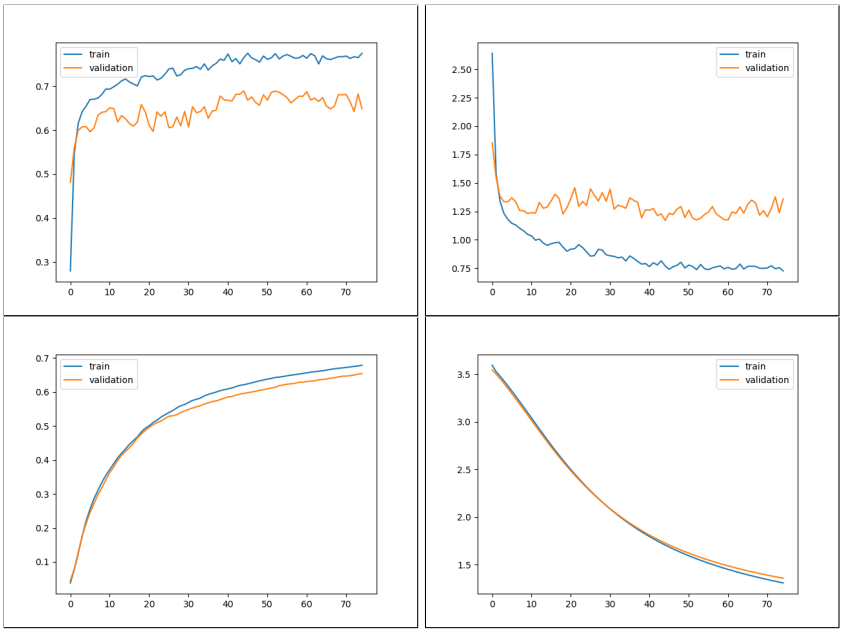
3.1

3.1.1





The accuracy obtained on the validation set is equal to 0.772.



3.1.2

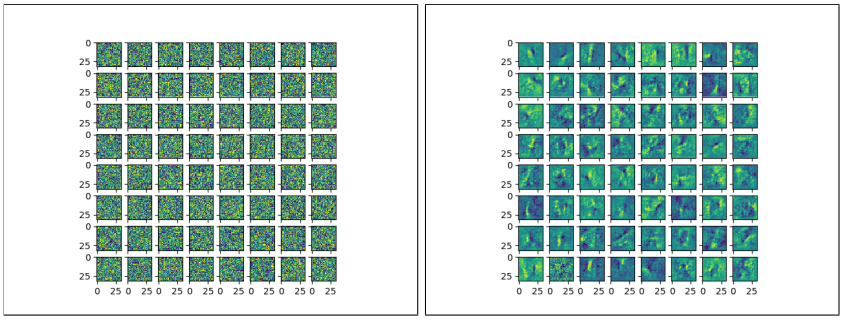
The first column shows the losses while the second column shows the accuracy plots. The first row shows the outputs for ten times the optimal learning rate while the second row shows the outputs for one-tenth the optimal learning rate. It can be seen that at a higher learning rate the losses are jittery. This is because the model keeps fluctuating between the optimal outputs.

At a lower learning rate, although the model tends to converge smoothly it never reaches its minima because the update after each iteration is very less.

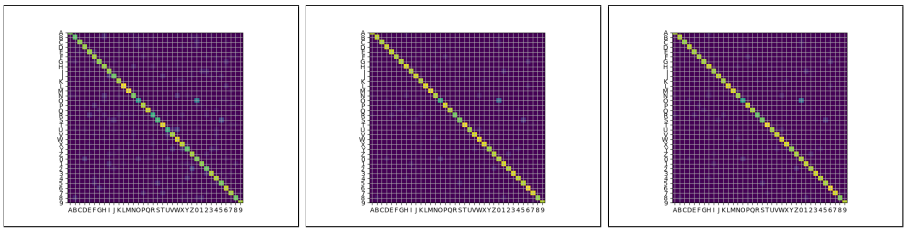
The test accuracy at the best model was 0.7728.

3.1.3

The initial weights are randomly initialized. The learned weights on the other hand are representative of the regions important for the text detection. The weights learned resemble edges of various orientations and scales.



3.1.4



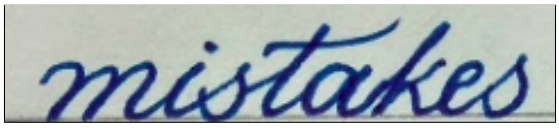
The confusion matrix is shows for the test, train and the validation columns in the first, second and third columns respectively. We can see that the classes getting most incorrect are pairs of ‘O’ and ‘0’ and ‘S’ and ‘5’.

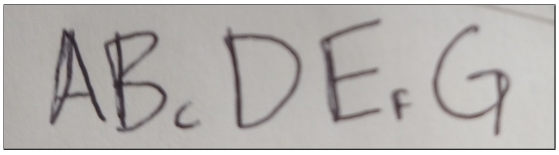
4 Extract Text from Images

4.1

The two assumptions that the sampling method makes are that :

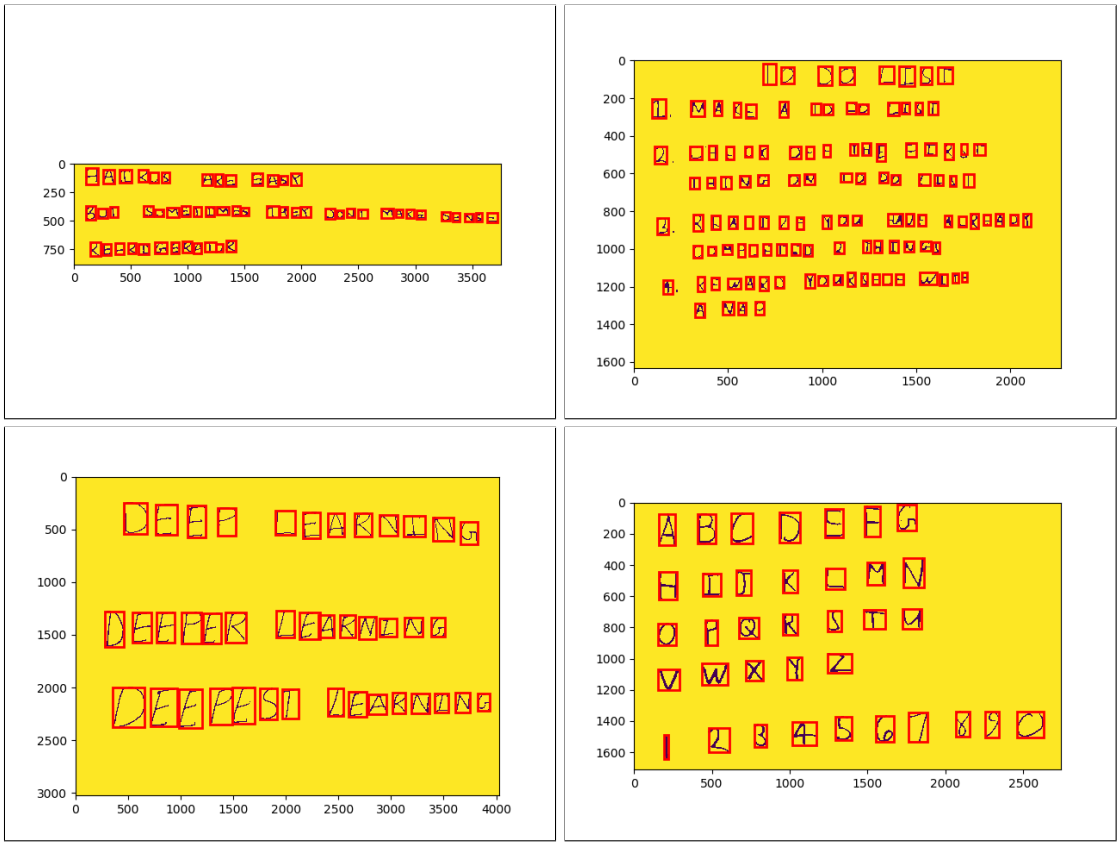
1. The letters are distinct. No one letter is connected to any other letter.
2. The letters are of the same size approximately.





4.2

4.3



4.4

TQDQLIST
IMAKEATDORLIST
2LHECKDFET3HEDFIRFT
TWXNGONQOLIST
BREALIZEYDWIHMIWVEGSALREADY
COMPLGTLDZ
4REWARDYDWRSELFWIIH
ANAP

HAIKWSAREHAGX
BWTSJMETIMESTHEXDDWTMAKOSHNQ6
RBFRIGBRATOR

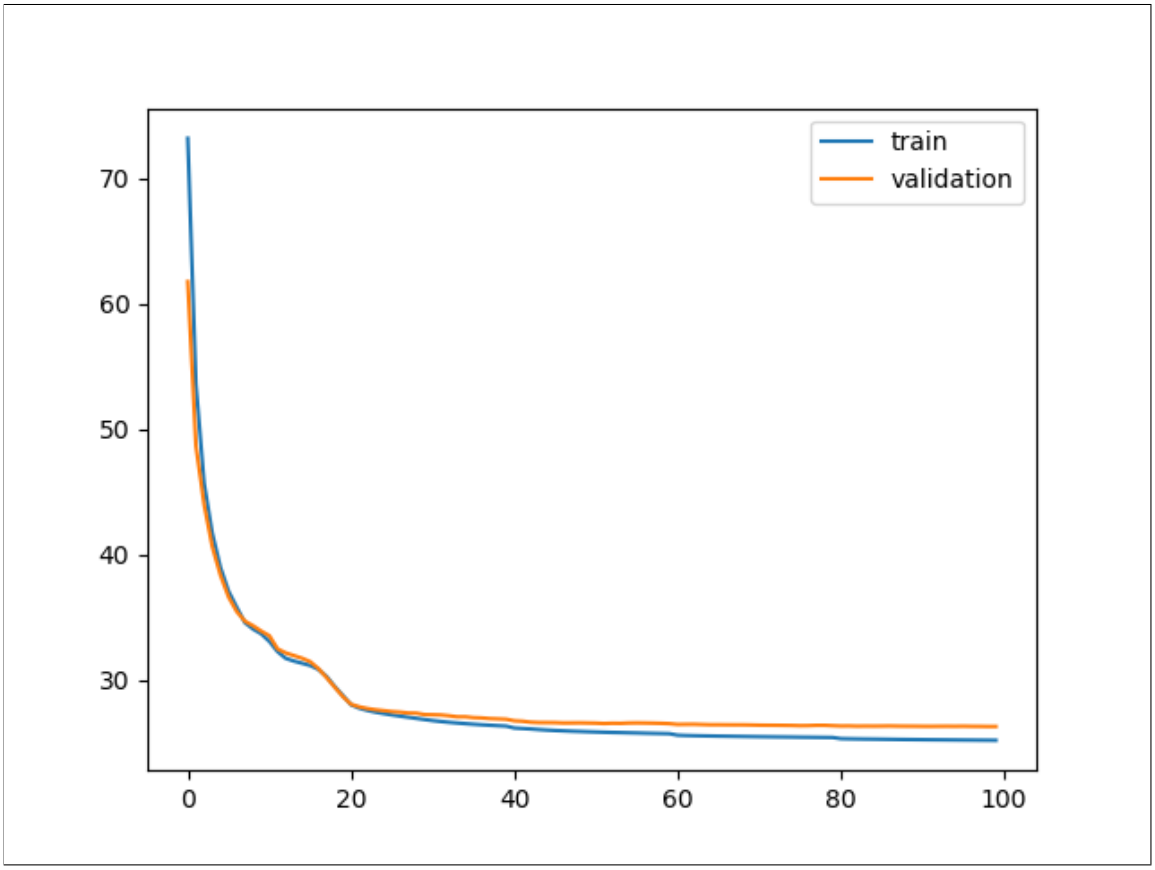
2BLDFFG
HIJKLMN
QPQRSTW
VWXYL
1Z3GSG7X93

DEEPLEARMLNG
DEPPEKLEAKNIMG
DF8P1S1LEARMING

5 Image Compression with Autoencoders

5.1

5.2



The loss decreases rapidly initially and then stabilizes even when the outputs are not exactly matching the input. The reason for this is the squared loss error being used.

5.3

5.3.1

The reconstructed validation images are broadly similar to the original inputs. However, the images are also blurred with excesive bleeding at the corners.

5.3.2

The PSNR obtained is 16.1023

6

6.1

The size of the projection matrix is 1024×32 with the rank equal to 32.

6.2

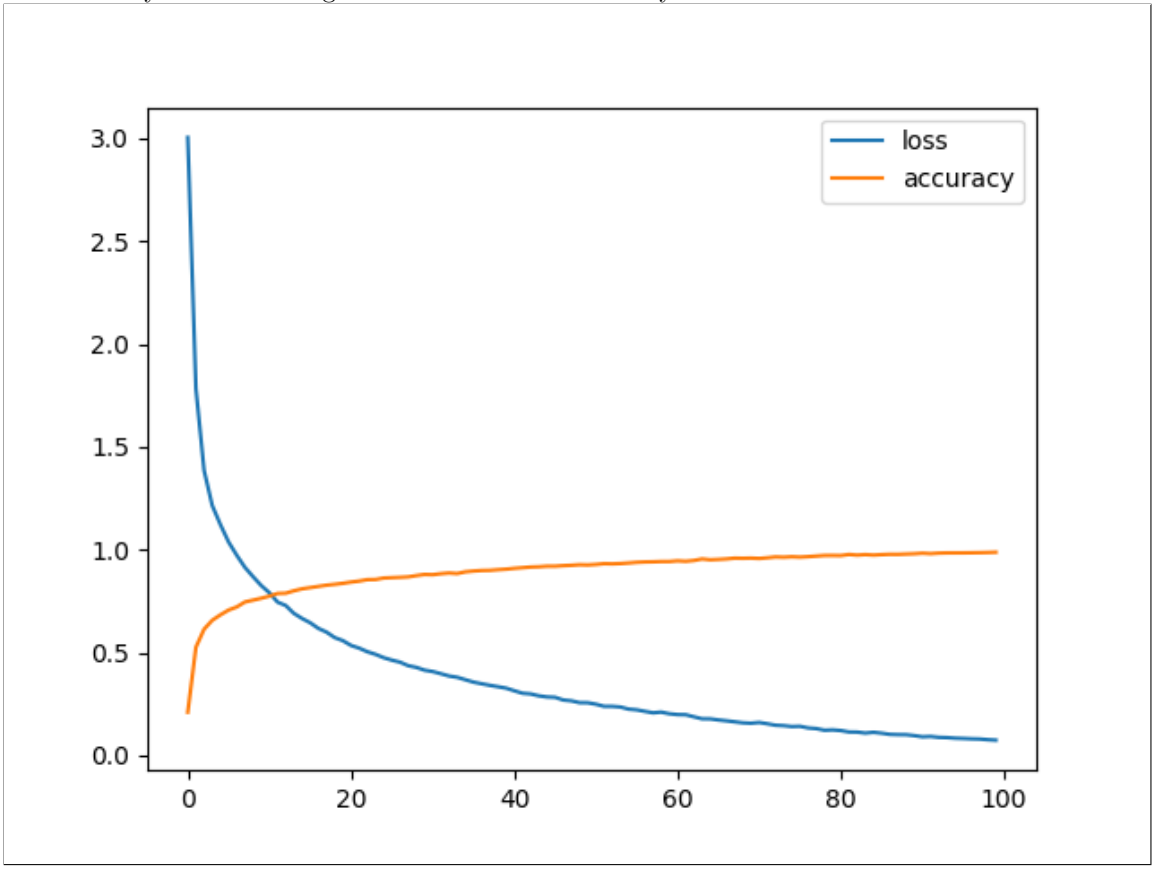
The PSNR obtained is 16.3484. The PSNR is slightly better for the autoencoder than the PCA because it just does not take the first n dimensions. The autoencoder on the other hand tends to combine all the dimensions to get the best output possible for the network.

7 PyTorch

7.1

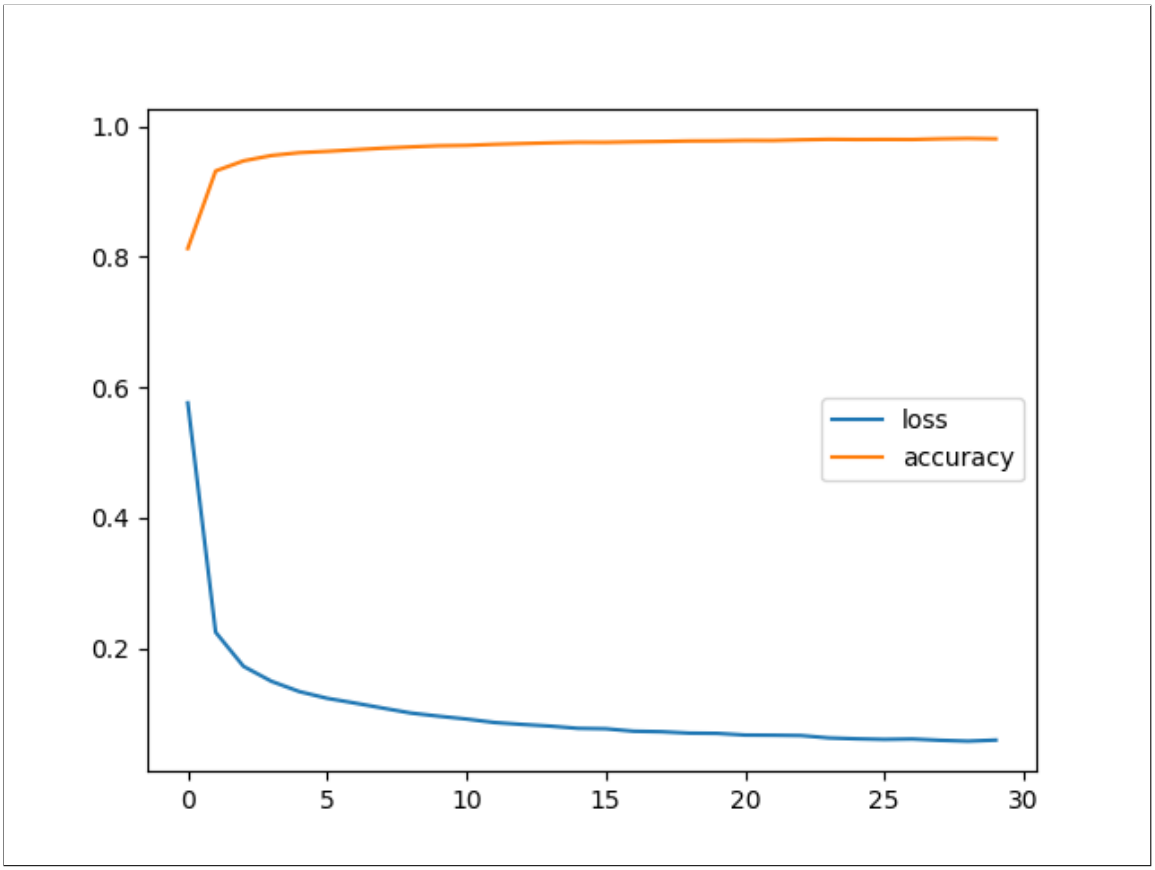
7.1.1

The accuracy on the training set is 0.98 while the accuracy on the test set is 0.80



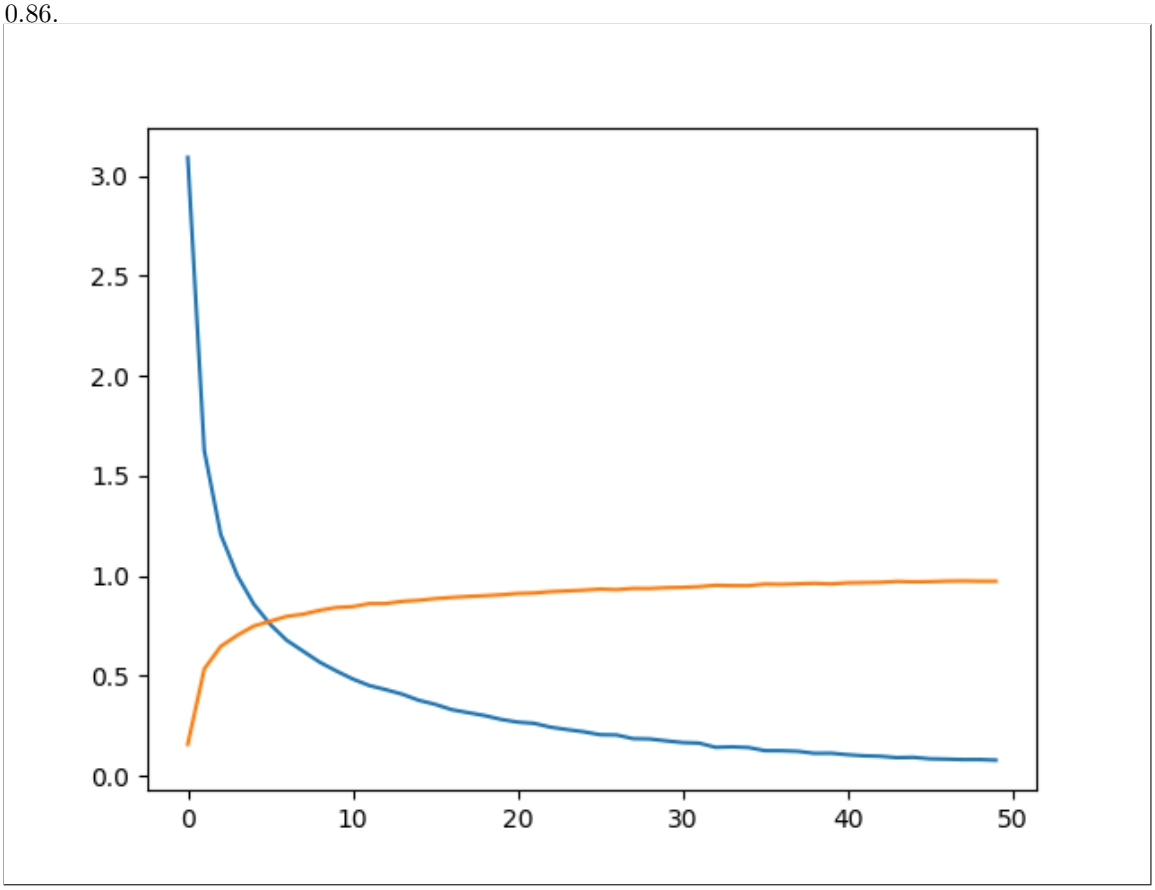
7.1.2

The accuracy on the training set is 0.9810 while the accuracy on the test set is 0.9803.



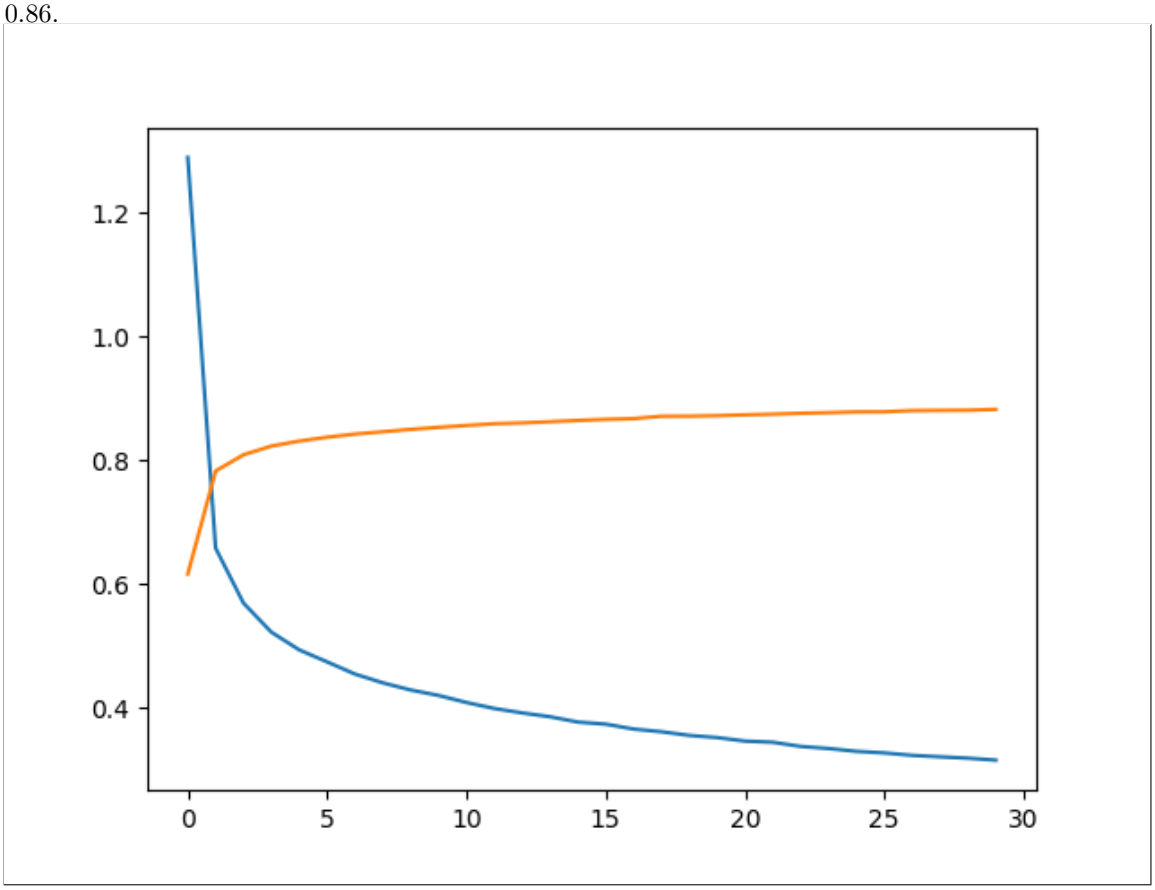
7.1.3

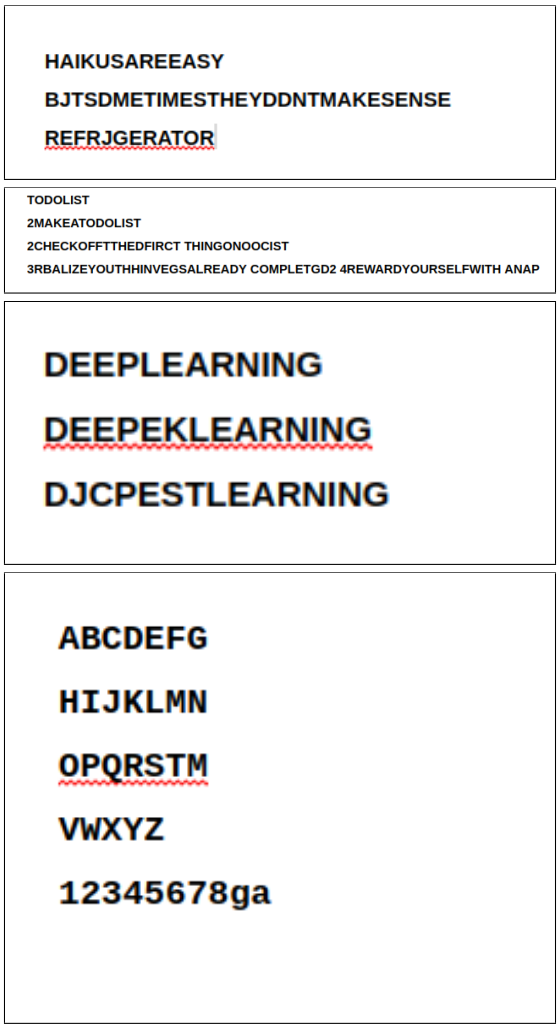
The accuracy on the training set is 0.9724 while the accuracy on the test set is 0.86.



7.1.4

The accuracy on the training set is 0.9724 while the accuracy on the test set is 0.86.




HAIKUSAREEASY
BJTSDMETIMESTHEYDDNTMAKESENSE
REFRJGERATOR
TODOLIST
2MAKEATODOLIST
2CHECKOFFTTHEDFIRCT THINGONOOCIST
3RBALIZEYOUTHHINVEGSALREADY COMPLETGD2 4REWARDYOURSELFWITH ANAP
DEEPLEARNING
DEEPEKLEARNING
DJCPESTLEARNING
ABCDEFG
HIJKLMN
OPQRSTM
VWXYZ
12345678ga

7.2

The accuracy on the training set is 0.46 and the accuracy on the test set is 0.41 when the model is trained from scratch. The graph for the same is shown on the left in the Figure.

The accuracy on the training set is 0.97 and the accuracy on the test set is 0.93 when the model weights are fine-tuned with the initialized weight from pretrained models is 0.93. The graph for the same is shown on the right in the Figure.