1 Q1.1

We have that softmax is defined as $softmax(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$
It follows that $softmax(x_i + c) = \frac{e^{x_i+c}}{\sum_j e^{x_j+c}}$

$= \frac{e^{x_i}e^c}{\sum_j e^{x_j}e^c}$
$= \frac{e^{x_i}}{\sum_j e^{x_j}} \frac{x^c}{x^c}$
$= \frac{e^{x_i}}{\sum_j e^{x_j}}$

It is often a good idea to use $c = -\max x_i$ so that $softmax(x)$ will never have an x value greater tha 0. This will mean that e^x will never be greater than one and thusly we don't have to worry about overflow as we would if x was too positive.

2 Q1.2

1. The range of each element in softmax is [0,1]
2. One could say that "softmax takes an arbitray real valued vector x and turns it into a probability vector."
3. Step $s_i = e^{x_i}$ values larger x_i more than smaller ones
   Step $S = \sum s_i$ determines the new vector interval
   Step $\frac{1}{S} s_i$ normalizes the vector

3 Q1.3

1. $\frac{\partial J}{\partial W} = \frac{\partial y}{\partial W} \cdot \frac{\partial J}{\partial y} = x \cdot \frac{\partial J}{\partial y}$
2. $\frac{\partial J}{\partial x} = \frac{\partial y}{\partial x} \cdot \frac{\partial J}{\partial y} = W \cdot \frac{\partial J}{\partial y}$
3. $\frac{\partial J}{\partial b} = \frac{\partial y}{\partial b} \cdot \frac{\partial J}{\partial y} = \frac{\partial J}{\partial y}$

4 Q1.4

1. $\begin{bmatrix} \mathrm{w}_0 & w_1 & w_2 & w_3 & w_4 & w_5 & w_6 & w_7 & w_8 \end{bmatrix} \begin{bmatrix} \mathrm{x}_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix} =$

   $\begin{bmatrix} \mathrm{w}_0 * x_0 + w_1 * x_1 + w_2 * x_2 + w_3 * x_3 + w_4 * x_4 + w_5 * x_5 + w_6 * x_6 + w_7 * x_7 + w_8 * x_8 \end{bmatrix}$

2. The dimensions are (64, 25) being multiplied by (25, 56)

Q2.1.1 Theory [2 points] Why is it not a good idea to initialize a network with all zeros? If you imagine that every layer has weights and biases, what can a zero-initialized network output after training?

The aim of weight initialization is to prevent layer activation outputs from exploding or vanishing during the course of a forward pass through a deep neural network. If either occurs, loss gradients will either be too large or too small to flow backwards beneficially, and the network will take longer to converge, if it is even able to do so at all.

Q2.1.3 Theory [1 points] Why do we initialize with random numbers? Why do we scale the initialization depending on layer size (see near Figure 6 in the paper)?

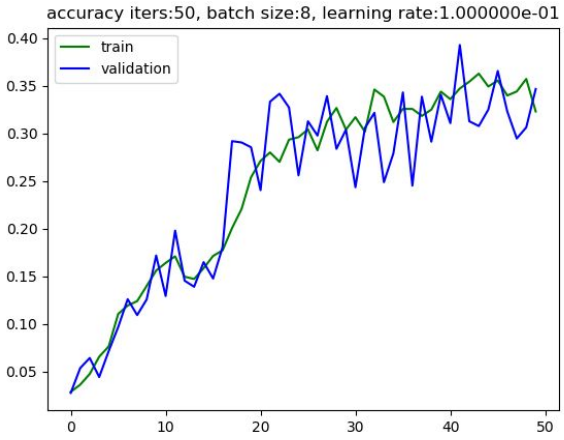
We initialize with random numbers to avoid the same pitfalls of initializing with all zeros.

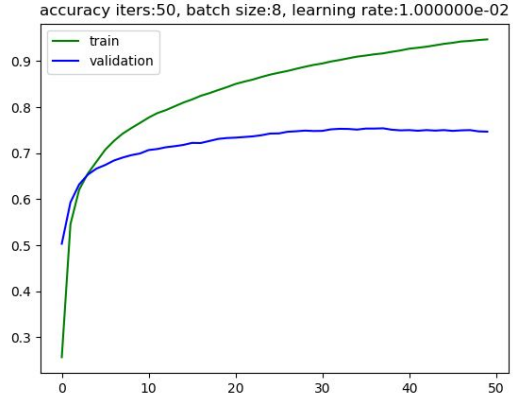
We scale the initialization as layer size dictates the number of things being multiplied together and if we do not scale it is possible that the product of multiplication on a large scale is too large for the computer to represent. Additionally a number that is too small can cause layer outputs to vanish

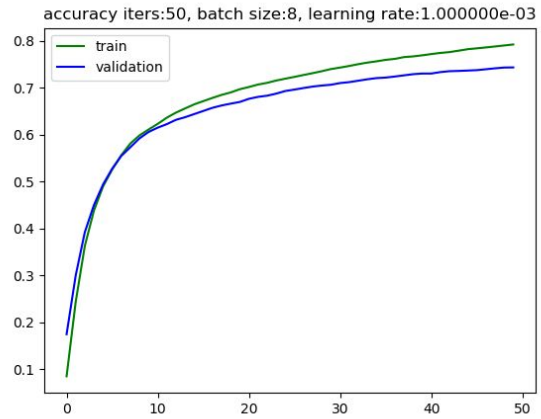
Q3.2 Writeup [3 points] Use your modified training script to train three networks, one with your best learning rate, one with 10 times that learning rate and one with one tenth that learning rate. Include all 4 plots in your writeup. Comment on how the learning rates affect the training, and report the final accuracy of the best network on the test set.

My ideal learning rate was 0.01. I found that if I went with anything an order of magnitude higher the gradients often had caused an over adjustment and I would end up with very volatile accuracies that would get better with one epoch and then get worse with the very next. On the opposite hand if I set my learning rate an order of magnitude lower it took far too long to traverse down the gradient descent and would take many more epochs to arrive at the same loss as my ideal learning rate.

Images below.

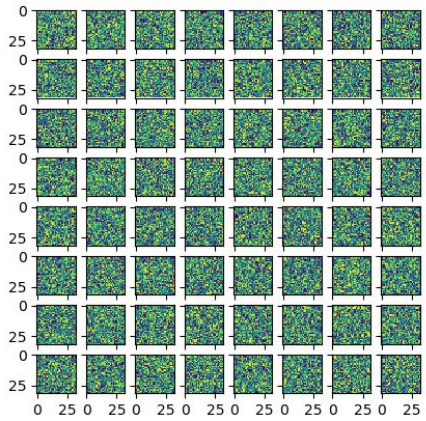

accuracy iters:50, batch size:8, learning rate:1.000000e-01
train
validation
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0
10
20
30
40
50


accuracy iters:50, batch size:8, learning rate:1.000000e-02
train
validation
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0
10
20
30
40
50

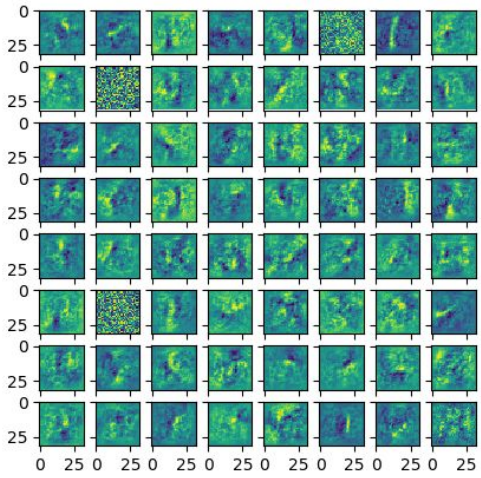

accuracy iters:50, batch size:8, learning rate:1.000000e-03
train
validation
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
10
20
30
40
50

Q 3.3

Initial



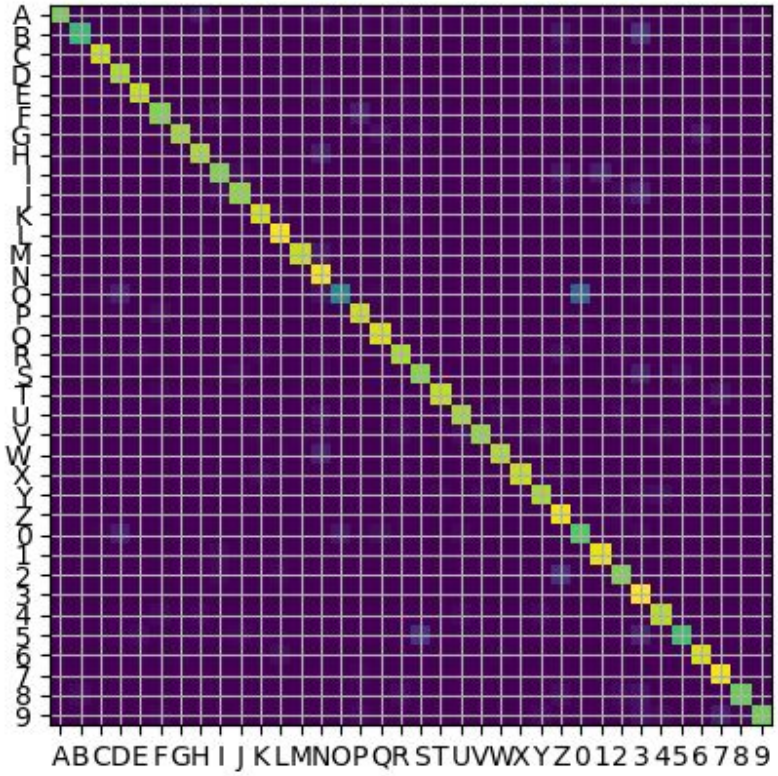
Network learned



As we can see whereas the initial weights appear as random static, the learned network appears to be detecting definite features with smooth transitions in and out of active regions.

Q 3.4

Most common confusion around s and 5, O and 0, and 2 and Z which makes sense as they appear the most similar.



Q 4.1

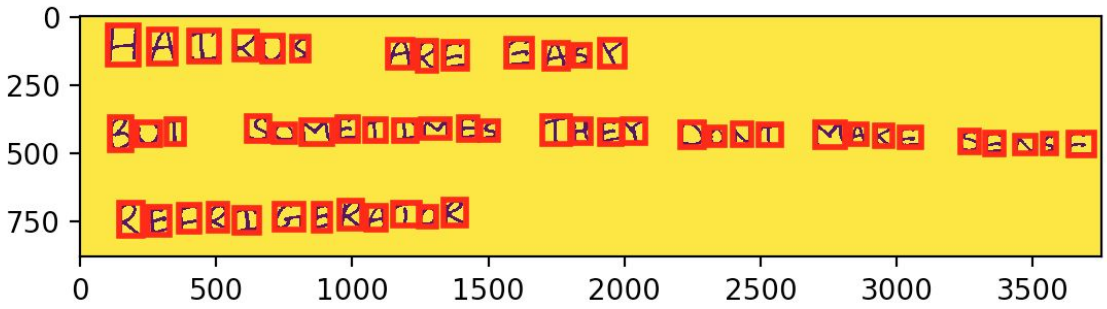
The method assumes consistent size of letters. If there was a disparity between sizes of letters it is likely that all but one set would be ignored.

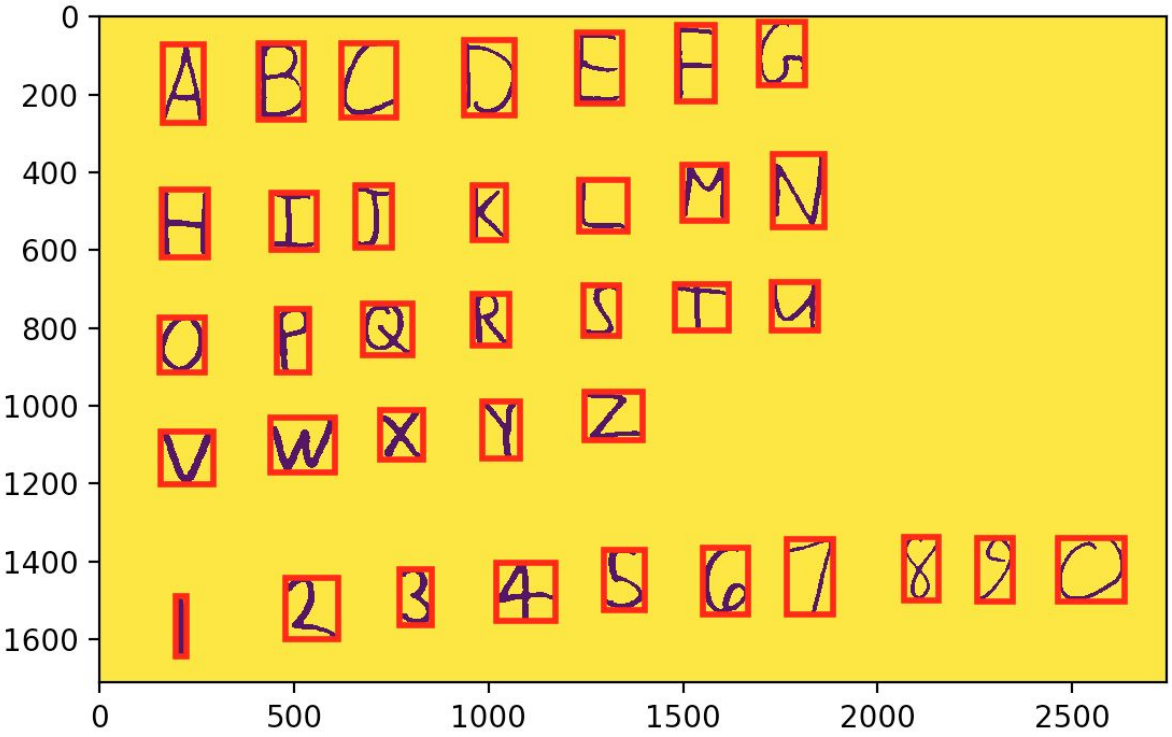
The method assumes zero rotation of the letters. The bounding boxes can only be inserted at a consistent angle such that if letters were rotated our classifier would likely fail terribly.

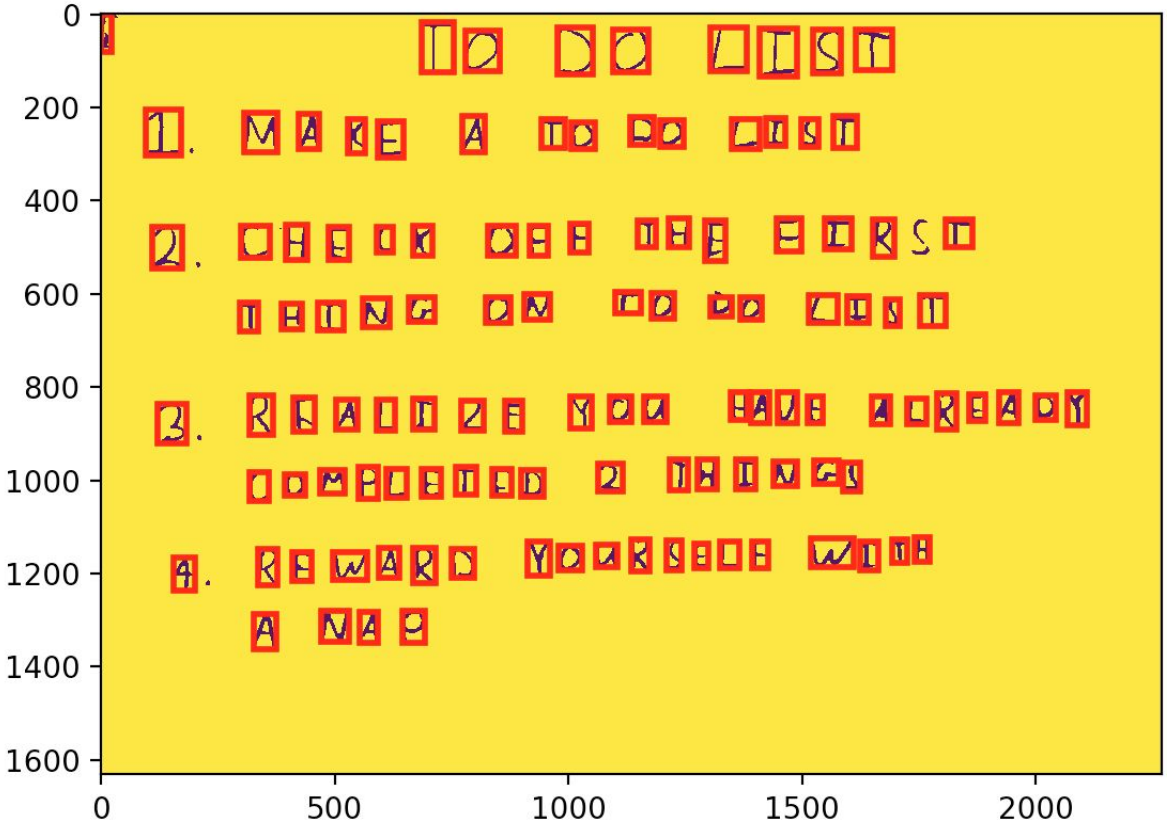
The method is likely to fail if characters are written with variance in size.

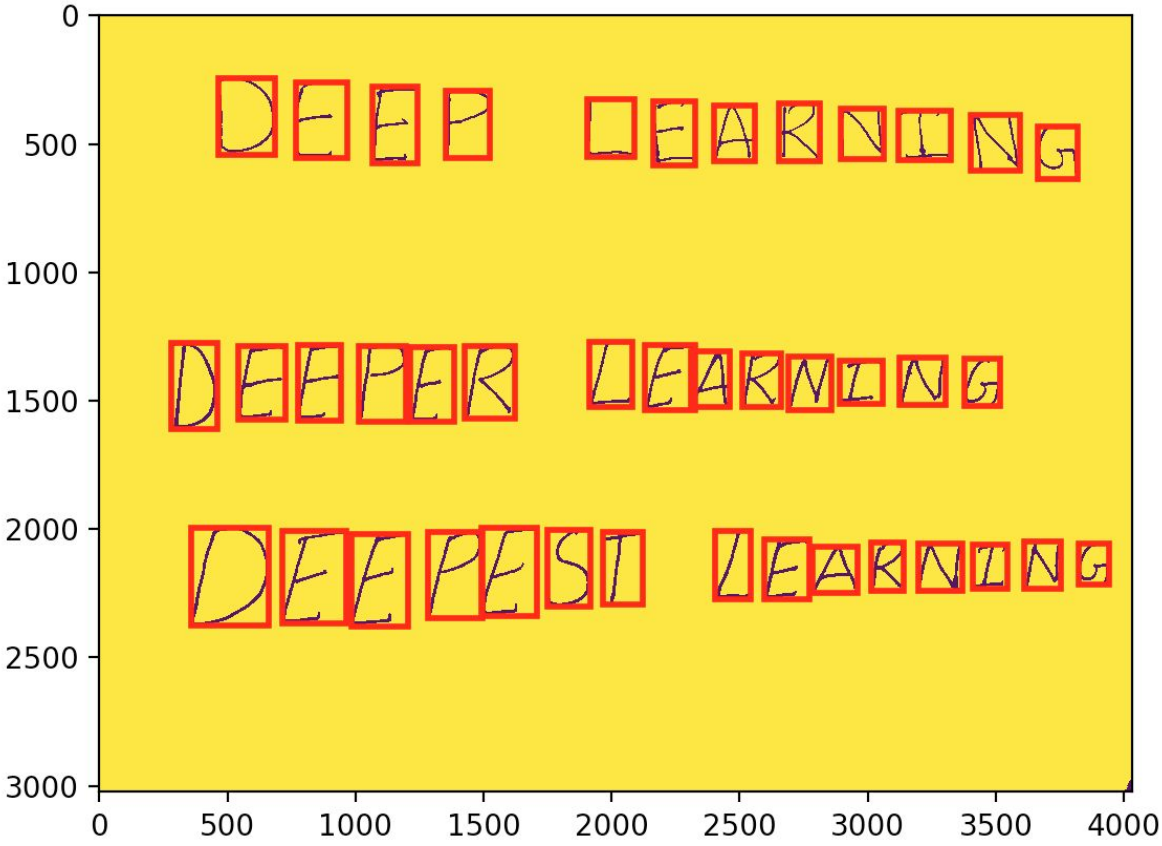
The method is likely to fail if characters are written non perpendicular to the image.

Q4.3


0
250
500
750
0
500
1000
1500
2000
2500
3000
3500


0
200
400
600
800
1000
1200
1400
1600
0
500
1000
1500
2000
2500


TO DO LIST
1. MAKE A TO DO LIST
2. CHECK OFF THE FIRST
THING ON TO DO LIST
3. REALIZE YOU HAVE ACHEAVED
COMPLETED 2 THINGS
4. REWARD YOURSELF WITH
A NAP
0
200
400
600
800
1000
1200
1400
1600
0
500
1000
1500
2000


DEEP LEARNING
DEEPER LEARNING
DEEPEST LEARNING
0
500
1000
1500
2000
2500
3000
3500
4000

Q 4.4

DEEP LEARKING

DEEPER LEARNING

DEEPESI LEARNING

F TQD0LIST

2 N2KE ATOQQ LIST

2 EH6CK OFF THE FIRST

THING QN TODOLIST

3 RIALIZE YQU HAVEA LR6ADT

CQMPL8TED Z YHINGS

4 R8WARD YOURSELF WITH

A NAP

ABCDEFG

HIJKLMN

QPQRSTU

VWXYZ

123GSG789O

HAIKUS ARE GASY

BLT SQMETIMEG THEY DDNT MAKE SENGE

REFRIGERATOR