Q1

16-720 HW#5 Conway Hsieh

Q1.1 Given $$\text{Softmax}(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

Then $$\text{Softmax}(x_i + c) = \frac{e^{x_i + c}}{\sum_j e^{x_j + c}}$$

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

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

$\therefore$ $\text{Softmax}(x_i) \equiv \text{Softmax}(x_i + c)$ $\forall c \in \mathbb{R}$, as desired.

If we set $c = -\max x_i$, when $x_i = \max$, we get $e^0 = 1$.
$\therefore$ Our range of values will between $(0, 1]$.
This will help prevent overflow issues when using translation.

Q1.2 1. The range of each element is $(0, 1]$
The ~~range~~ sum of all elements is 1

2. One could say that "softmax takes an arbitrary real valued vector x and transit into a probability distribution, where softmax(i) gives the probability of getting softmax(x_i)

3. ① $s_i = e^{x_i}$, calculates probability of x_i in exponential
② $S = \sum s_i$ totals the outcome frequency
③ $\frac{1}{S} s_i$ normalizes each value

Q1.3 ~~Given $\sigma(x) = \frac{1}{1+e^{-x}}$~~

Multi-layer Network w/ Linear activation function:

$y = W_n x_n + b_n$

$= W_n (W_{n-1} x_{n-1} + b_{n-1}) b_n$

Rearrage, we get

$y = W_n W_{n-1} x_{n-1} + W_n b_{n-1} + b_n$

$= W' x_{n-1} + b'$

If you extrapolate for further layers

$y = W' (W_{n-2} x_{n-2} + b_{n-2}) + b'$

which can be reduced to

$y = Wx + b$, which is linear

Q1.4 Given $\sigma(x) = \frac{1}{1+e^{-x}}$

$\nabla(\sigma(x)) = \frac{d\sigma(x)}{dx}$

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

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

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

Q1.5 Given $y = x^T W + b$ (or $y_i = \sum_{i=1}^{d} x_i W_{ij} + b_j$)

$\frac{\delta J}{\delta W} = \frac{\delta J}{\delta y} \frac{\delta y}{\delta W}$, $\frac{\delta J}{\delta y} = \delta \in \mathbb{R}^{k \times 1}$

$\frac{\delta J}{\delta W} = \delta x$, $\frac{\delta J}{\delta W_{ij}} = \delta_j x_i$

$\frac{\delta J}{\delta x_i} = \frac{\delta J}{\delta y_j} \frac{\delta y_j}{\delta x_i} = \sum_j W_{ij} \delta_j$

$\frac{\delta J}{\delta b_j} = \frac{\delta J}{\delta y_j} \frac{\delta y_j}{\delta b_j} = \delta_j$

Reforming Matrices

$\frac{\delta J}{\delta W} = x \delta^T$, $\frac{\delta J}{dx} = W \delta$, $\frac{\delta J}{\delta b} = \delta$

Q1.6 ① Using a sigmoid function for many layers may lead to a vanishing gradient because:
The derivative of sigmoid, its range is only $[0, 0.25]$, which significantly reduces any number. Used for successive layers, the gradient will continue to shrink.

② tanh output range: $[-1, 1]$ vs sigmoid $[0, 1]$
tanh is preferable because it maps positive values to positive values, and negative values to negative ones

③ The derivative of tanh has a range of $[0, 1]$.
Therefore, it doesn't shrink as fast as the derivative of a sigmoid would

④ Given $\tanh(x) = (1 - e^{-2x})/(1 + e^{-2x})$

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

$\tanh(x) + 1 = 2\sigma(2x)$

$\therefore \tanh(x) = 2\sigma(2x) - 1$ as desired.

Q 2.1.1

If the weights are initialized to zero, whenever you try to propagate, you multiply weights by delta, resulting in net zero change in weights. Therefore, a zero-initialized network can only output zeros, and never learn anything.

Q 2.1.3

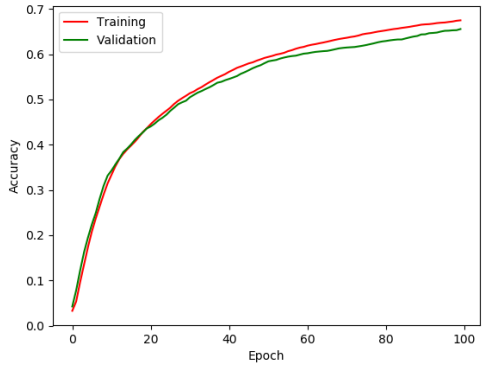
Initialization is done with random numbers allows the network to probe different parts of the solution space to find the best solution. If you always start with the same weights, the training and gradients will remain similar, limiting the scope of your solution.

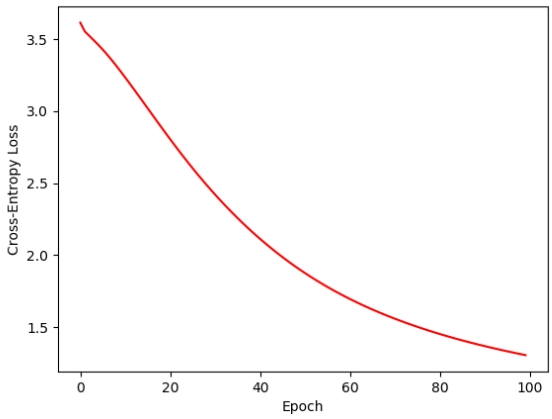
Scaling the initialization based on layer size is done because when consecutive layers have the same dimension, the average activation variance that is conserved increases. This allows for more information to continue through the network.

Q 3.1.2

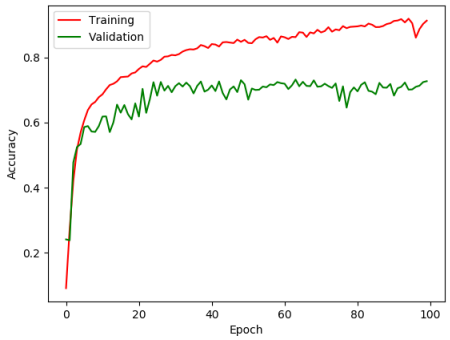
Final validation accuracy of best set: 76%

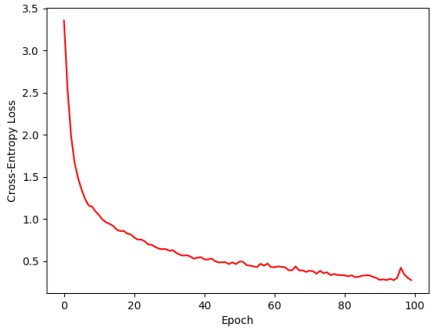
10x less learning rate:





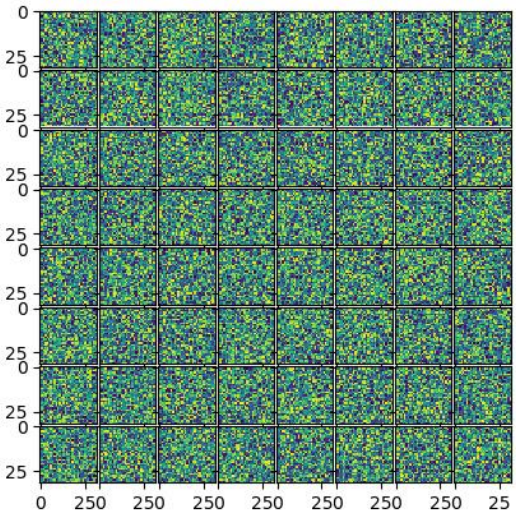
10x more learning rate:



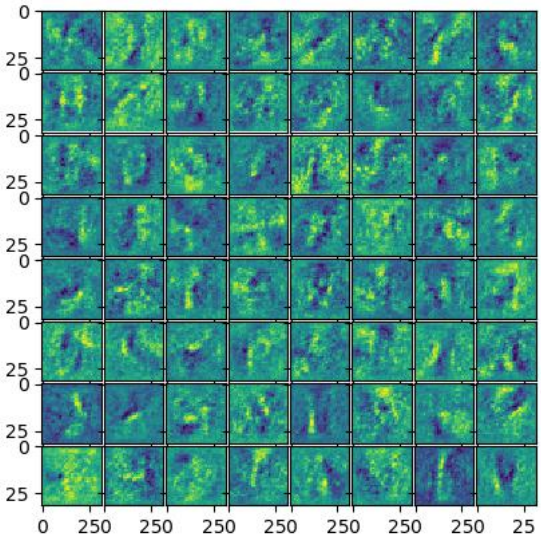


More learning rate causes more abrupt, jagged changes to weights/ training while less learning rate is smooth. Also, achieves less accuracy in same number of iterations (because learns more slowly)

Q3.1.3



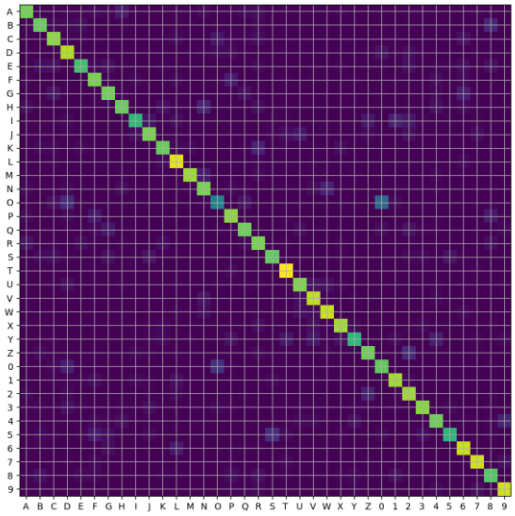
Initialization



After Training

Initialized weights look random, as expected. After training, structures begin to emerge within the weights, which do not look like noise as it did at initialization.

Q 3.1.4



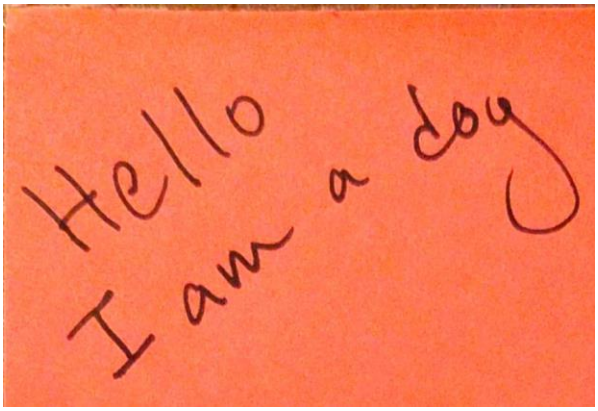
O and 0 seems to be the most prominent case of misclassification. Others include 5 and S, 2 and Z, and Y and 4. These seem reasonable, as due to differences in handwriting, they have similar structures.

Q4.1

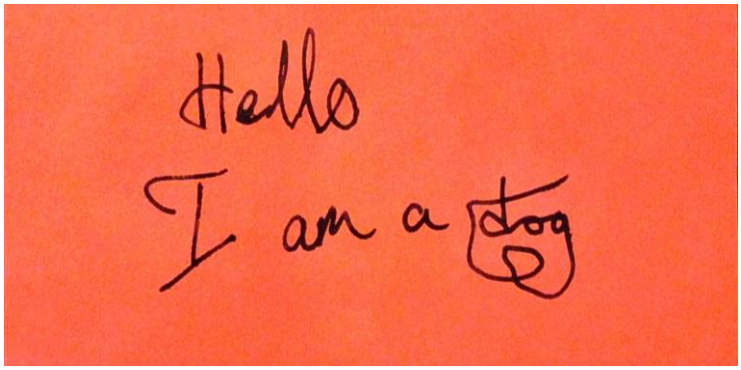
One assumption is that the words are in left to right, top to bottom format. This means that if any rotations are made, the classification order will be incorrect.

Another assumption is that all the letters are fully connected when written, and separate letters are not connected.

Lastly, an assumption is that all letters are of similar size, so anything too small will be considered as noise.

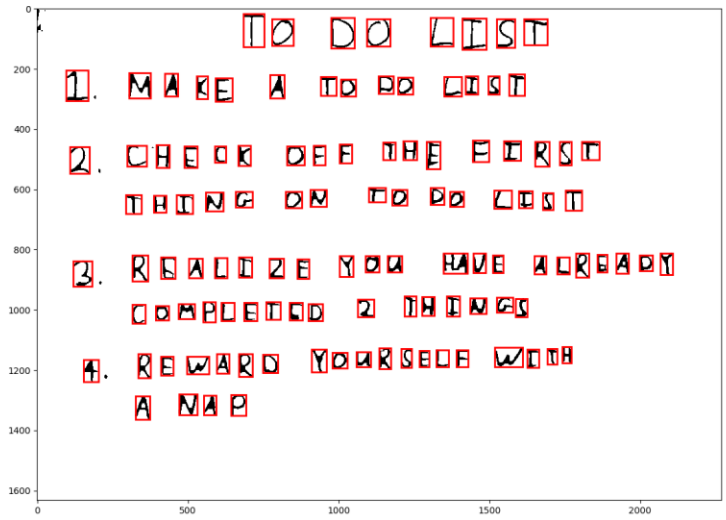


Rotation will cause issues during classification

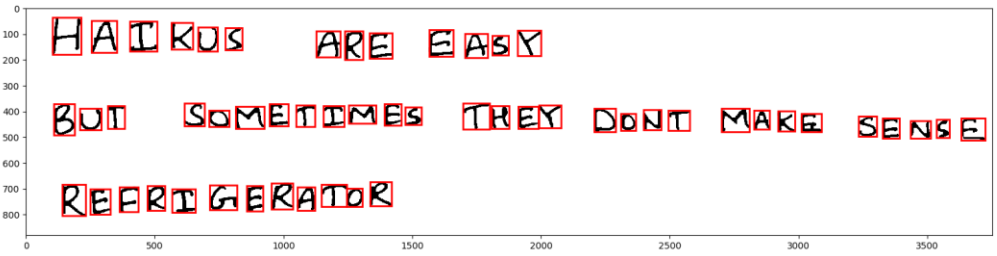


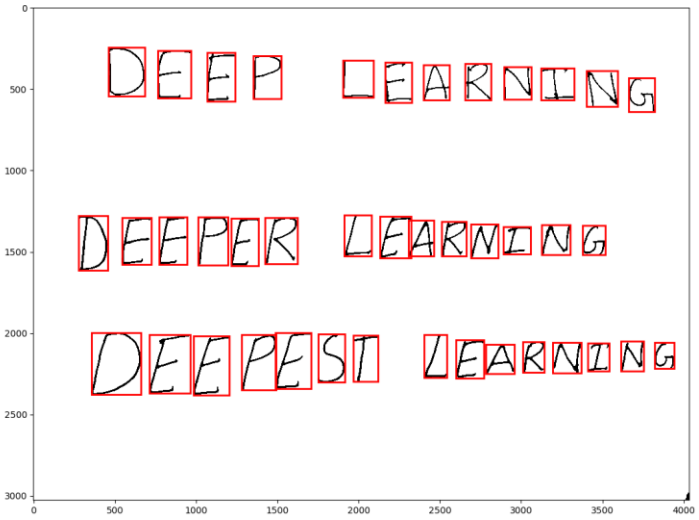
Connected letters will cause issues

Q 4.3


TO DO LIST
1. MAKE A TO DO LIST
2. CHECK OFF THE FIRST THING ON TO DO LIST
3. REALIZE YOU HAVE ALREADY COMPLETED 2 THINGS
4. REWARD YOURSELF WITH A NAP


A B C D E F G
H I J K L M N
O P Q R S T U
V W X Y Z
1 2 3 4 5 6 7 8 9 0


HAIKUS ARE EASY
BUT SOMETIMES THEY DONT MAKE SENSE
REERT GERATOR


DEEP LEARNING
DEEPER LEARNING
DEEPEST LEARNING

Q 4.4

F0 D0 LI5T

I MAKE R T0 P0 LI5T

2 CH5CK 0FR THR TIRST THING 0N T0 D0 LI5T

3 RRALIZE YO4 M4RALRE A0T COMPLETRD 2 THIN4S

4 REWARD F0URSELR WITR A NRP

ABCDEFG

HIJKLMN

0PQRST4

VWXYZ

1Z345G787Q

HAIKU5 ARR EA5Y

BUF SOMRFIMRS TRRY 0ONT MAKR SR45R

RRFRI6ERAT0R

DEEP LEARMINQ

DEEFER LEARN2NG

QEEPE5T LEARNING