Word Recognition HW

**Original Positive Words**

To start with, I take 7 random positive words from website:

<https://www.randomlists.com/random-words?dup=false&qty=20>

Those words came out as:

**['nice', 'toothpaste', 'highfalutin', 'childlike', 'authority', 'halting', 'found']**

As seen above, they have different length, so we use windowing techniqueto represent all words at same length. How we do is by first finding the word with longest length (highfalutin = 11 letters) then adding 1, so our window length becomes 12. After this we start padding spaces to all other words with shorter length. As seen below all padded words now have length of 12:

'nice ', 'toothpaste ', 'highfalutin ' **…** 'found '

**Creating Training Data**

We need to create training input (size of 10,000 words) out of original 7 words. Training input will contain both positive words and negative words. We create itby randomly shuffling the 0-3 letters of original 7 positive words. After that, we pad the spaces to words using windowing technique. Note that approximately 55% of these 10,000 words inside training data are positive, the rest are negative.

Then we create training targets which are correct outputs for given inputs**.** We encode the output as a vector of binaries where "1" will be in ith position if the model recognize the input word as ith word from original positive list of 7 words. For example, if we give word “nice” to the network, the network should produce [1, 0, 0, 0, 0, 0, 0] indicating that word is 0th word out of 7 positive words.

**Encoding Training Data**

Then as we have to encode the input words as binary rather than letters, next step is to represent each letter of word in binary form. We encode each letter with 5 bits, so by this way we convert input words of length 12 into length 60;

' ' = 00000 (0)

'a' = 00001 (1)

...

'z' = 11010 (26)

**Network Specifications**

Layers size: 60 – 10 – 10 – 7

(60 input layer neurons, 10 first hidden layer neurons, 10 second hidden layer neurons, 7 output layer neurons)

Activation function: ReLU

Solver: Adam

Batch size: 300

Learning Rate: 0.001

Iterations: 200 with possibility of early stopping.