HOMEWORK 3

CMU 10-707: DEEP LEARNING (FALL 2017)
https://piazza.com/cmu/fall2017/10707
OUT: Nov 1
DUE: Nov 15
TAs: Dimitris Konomis, Dheeraj Rajagopal, Yao-Hung (Hubert) Tsai

1 Problem 1 (20 pts)

Forward Propogation

$$\begin{aligned}
\tilde{\mathbf{x_j}} &\in \mathbb{R}^{8000} \\
z_{j,k} = \vec{x_j} * \vec{\theta_k}, \quad \tilde{\theta_\mathbf{k}} &\in \mathbb{R}^{D} \\
Z_l &= z_{l/D, l\%D} \\
h_h = \vec{\delta_h} * \vec{Z} + a_h, \quad \tilde{\delta_\mathbf{h}} &\in \mathbb{R}^{3*D} \\
H &= tanh(h) \\
p_v = exp(\vec{\beta_v} * \vec{H_v} + b_v), \quad \tilde{\beta_\mathbf{v}} &\in \mathbb{R}^{H} \\
y_v = P(w_i = v / w_{i-1}, w_{i-2}, w_{i-3})) & \\
y_v &= \frac{p_v}{\sum_v p_v} \\
J &= -log(y_v)
\end{aligned}$$

Dimensionality of input one hot vector and all parameters

$$\begin{aligned}
\tilde{\mathbf{x_j}} &\in \mathbb{R}^{8000} \\
\tilde{\theta} &\in \mathbb{R}^{D*8000} \\
\tilde{\delta} &\in \mathbb{R}^{3*D*H} \\
\tilde{\mathbf{a}} &\in \mathbb{R}^{H} \\
\tilde{\beta} &\in \mathbb{R}^{H*V} \\
\tilde{\mathbf{b}} &\in \mathbb{R}^{V}
\end{aligned}$$

(Note that the softmax can also be represented as H*(V-1) weights if the normalization is implicit)

Backward Propogation

$$\frac{dJ}{dy} = [0, 0, ..., (-1/y_v), ..., 0, 0]$$
$$\frac{dy_{v1}}{dp_{v2}} = y_{v1} * (\mathbb{1}_{v1,v2} - y_{v2})$$
$$\frac{dp_v}{db_v} = 1$$
$$\frac{dp_v}{d\beta_{hv}} = H_h$$
$$\frac{dp_v}{dH_h} = \beta_{hv}$$
$$\frac{dH_h}{dh_h} = 1 - (tanh(h_h))^2$$
$$\frac{dh_h}{da_h} = 1$$
$$\frac{dh_h}{d\delta_{l,h}} = Z_l$$
$$\frac{dh_h}{dZ_l} = \delta_{l,h}$$
$$\frac{dZ_l}{dz_{j,k}} = \mathbb{1}_{(j,k)\equiv(l/D, l\%D)}$$
$$\frac{dz_{j,k}}{d\theta_{k,q}} = x_{j,q}$$

2 Problem 2 (20 pts)

How many gates are there in LSTM and GRU, respectively? Please also specify the names of gates in LSTM/GRU.
LSTM has 3 gates: Forget(f), Input(i) and Output(o).
GRU has 2 gates: Update(z) and Reset(r).

Summarize in your own words, the functions of the gates in the GRU and LSTM, and then compare between them.

LSTM

$$\mathbf{f}_t = \sigma\left(W_f \mathbf{x}_t + U_f \mathbf{h}_{t-1} + b_f\right)$$
$$\mathbf{i}_t = \sigma\left(W_i \mathbf{x}_t + U_i \mathbf{h}_{t-1} + b_i\right)$$
$$\mathbf{o}_t = \sigma\left(W_o \mathbf{x}_t + U_o \mathbf{h}_{t-1} + b_o\right)$$

Forget gate: Moderates what part of previous cell state to forget based on the previous output and current input. Example: The cell state keeps track of the gender of the subject in a text, But it needs to forget the gender if it encounters that the text introduced a new subject.
Input gate: Moderates what part of current inputs to incorporate into the memory cell. Example: An input word such as ”he” or ”she” should used to incorporate the gender of the subject in the cell state since the network may need it to output the correct adjective at a future time say ”handsome” for male or ”beautiful” for female.
Output gate: Moderates what part of cell state to use for the current output. Example: While the gender information may be a critical part of the cell state, it may not be useful to predict the word right away.

GRU

$$\mathbf{z}_t = \sigma\left(W_z\mathbf{x}_t + U_z\mathbf{h}_{t-1} + b_z\right)$$
$$\mathbf{r}_t = \sigma\left(W_r\mathbf{x}_t + U_r\mathbf{h}_{t-1} + b_r\right)$$

The LSTM uses the same variables x_t, h_{t-1} to calculate input (i_t) and forget (f_t) gate. This is redundant. The GRU simply combines these function by using a single update (z_t) gate instead of two.
LSTM has a redundant non linearity i.e. when calculating $\tilde{\mathbf{c}}_t$ and $tanh(c_t)$. The GRU also does not maintain a seperate cell state. It uses the reset gate r_t to moderate between using the current input and previous output to calculate the current output.

In terms of outputs, what are the differences between LSTM and GRU?

LSTM

$$\tilde{\mathbf{c}}_t = \tanh\left(W_c\mathbf{x}_t + U_c\mathbf{h}_{t-1} + b_c\right)$$
$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh\left(\mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t\right),$$

GRU

$$\tilde{\mathbf{h}}_t = \tanh\left(W\mathbf{x}_t + U(\mathbf{r}_t \odot \mathbf{h}_{t-1}) + b\right)$$
$$\mathbf{h}_t = \mathbf{z}_t \odot \tilde{\mathbf{h}}_t + (\mathbf{1} - \mathbf{z}_t) \odot \mathbf{h}_{t-1}.$$

The LSTM uses the same variables x_t, h_{t-1} to calculate input (i_t) and forget (f_t) gate. This is redundant. The GRU simply combines these function by using a single update (z_t) gate instead of two.
LSTM has a redundant non linearity i.e. when calculating $\tilde{\mathbf{c}}_t$ and $tanh(c_t)$. The GRU also does not maintain a seperate cell state.
The GRU output at a given state is therefore all that it passes over to the next state.

Suppose $\mathbf{x} \in \mathbb{R}^m$ and $\mathbf{h} \in \mathbb{R}^n$ ($\mathbf{c} \in \mathbb{R}^n$), please compute the numbers of parameters in each LSTM and GRU unit, respectively.
LSTM: $4 * (mn + n^2 + n)$
LSTM: $3 * (mn + n^2 + n)$

Suppose you have two networks consisting of a sequence of LSTM or GRU cells (with the same number of cells). Which networks might take less time to train and generalize? Please also explain why in a few sentences.
Given the same amount of data and same number of cells the GRU will train faster and generalize better since it has fewer paramters.
However in certain cases where the specific extra model complexity of LSTM is necessary to learn long term dependencies, the GRU may not converge to a minimum training error at all. In such cases given enough data, the LSTM will outperform GRU for accuracy on both training and test set.

3 Problem 3 (60 pts)

Using the standard SGD on the entire dataset was frequently getting stuck in local minima i.e. the model learns to predict only the top 2-4 most frequent words resulting in an accuracy of $6 - 7\%$. Therefore I use the following approach to significantly speed up the learning for all sections of problem 3.

- Train by excluding all 4 grams where the 4th word is one of 10 most frequent words.
- Train further by excluding all 4 grams where the 4th word is one of 5 most frequent words.
- Train further by excluding all 4 grams where the 4th word is one of 2 most frequent words.

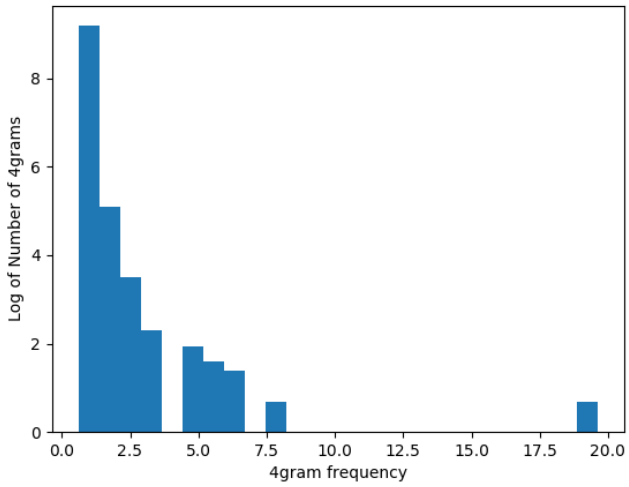

Figure 1: Four gram distribution

- Train further by including all 4 grams.

This approach does slightly bias the model in favour of less frequent 4th words among all 4-grams. This issue is minimized by performing fewer epochs for step 1,2,3 (say 2-5 epochs) and larger epochs of step 4 (say 50-100 epochs). Typically this approach converges much faster than the standard SGD.

(Note that the entire validation set without any exclusion is used throughout. Therefore the comparison between validation and train curves would typically show a large gap which might misleading indication of low generalization.)

3.1 Data Preprocessing (10 pts)

How many trainable parameters are in the network including the weights and biases ?

- Embedding Layer: 8000x16
- Hidden Layer: 48x16 + 16 = 784
- Softmax: 7999*16 + 7999 = 135983
- Total parameters: 264767

The most common 4-grams (top 50): [0] $ UNK million *u* [1] UNK UNK . END [2] 0 *t*-1 . END [3] , said 0 it [4] million *u* . END [5] $ UNK billion *u* [6] says *t*-1 . END [7] $ UNK *u* a [8] million *u* , or [9] UNK *u* a share [10] or $ UNK *u* [11] said 0 *t*-1 . [12] the UNK of the [13] , or $ UNK [14] *u* , or $ [15] the national association of [16] 0 *t*-2 . END [17] year earlier . END [18] and UNK . END [19] UNK million *u* , [20] of $ UNK million [21] of UNK . END [22] START the company ” s”[23] , ” ” says *t*-1 [24] to UNK . END [25] the UNK . END [26] , according to the [27] $ 130 million *u* [28] does ”n t” already own [29] UNK million *u* from [30] $ UNK *u* in [31] UNK million *u* . [32] 0 it does ”n t”[33] START a spokesman for [34] UNK *-1 . END [35] *-1 to close at [36] *u* a share , [37]” ” says *t*-1 UNK [38] for the UNK UNK [39] UNK . ” ” END [40] it does ”n t” already [41] UNK , conn. , [42] $ UNK *u* , [43] the company said 0 [44] UNK UNK , a [45] UNK *t*-2 . END [46] UNK *u* . END [47] *u* a share . [48] UNK $ UNK million [49] a share . END [50]

(Note that some end of line or special characters have been removed on latex, but are part of the training corpus)

(Figure 1) A large number of 4grams appear only one. Only a handful of four grams appear more frequently say 5 times or more. Therefore its unlikely that we will observe same 4 grams across training and test set.

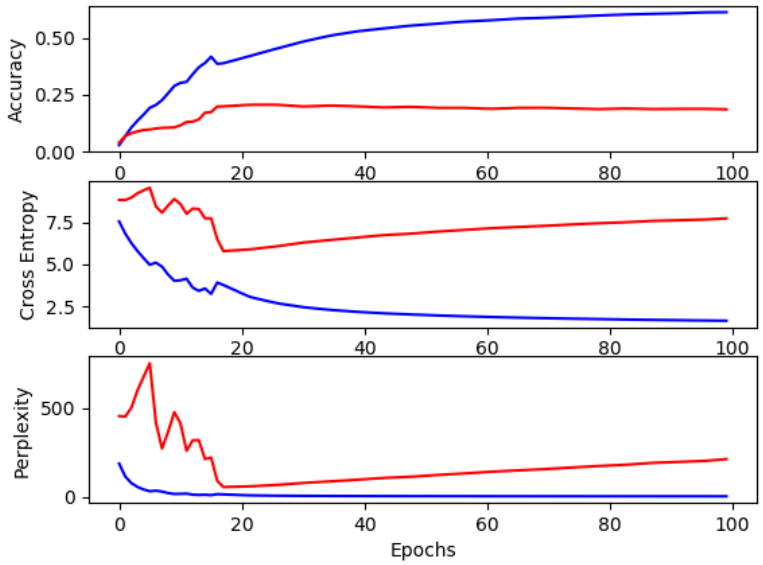

Figure 2: 128 Hidden Units, Linear Activation.

3.2 Backpropagation with Linear Hidden Layer (20 pts)

See Figure 2
Best performance achieved Perplexity: 51.98 or (295.7 base 2)
Cross Entropy: 5.7
Accuracy: 21.1% (Percentage of 4 grams where the right word was predicted to have the maximum probability)

See Figure 3
The network with 256 hidden units does slighly better on the validation set compared to 128 or 512 hidden units.

3.3 Incorporating Non-linearity (10 pts)

See Figure 4
Network with tanh activation (thicker curve) performs better than linear activation.

3.4 Analysis (10 pts)

5 sample sentences generated using 3 initial words.

- government of united states . END , the company said 0
- he says to a lack of enthusiasm with the latest numbers
- life in the officials said 0 they were getting out of
- city of new orleans , $ 7,500 *u* fine and one-week
- a spokesman for the irs confirmed that " there is finally

5 sample words and their closest neighbours from a sample of 1000 most frequent words.

- 'days': 'months', 'time', 'stocks', 'expected'

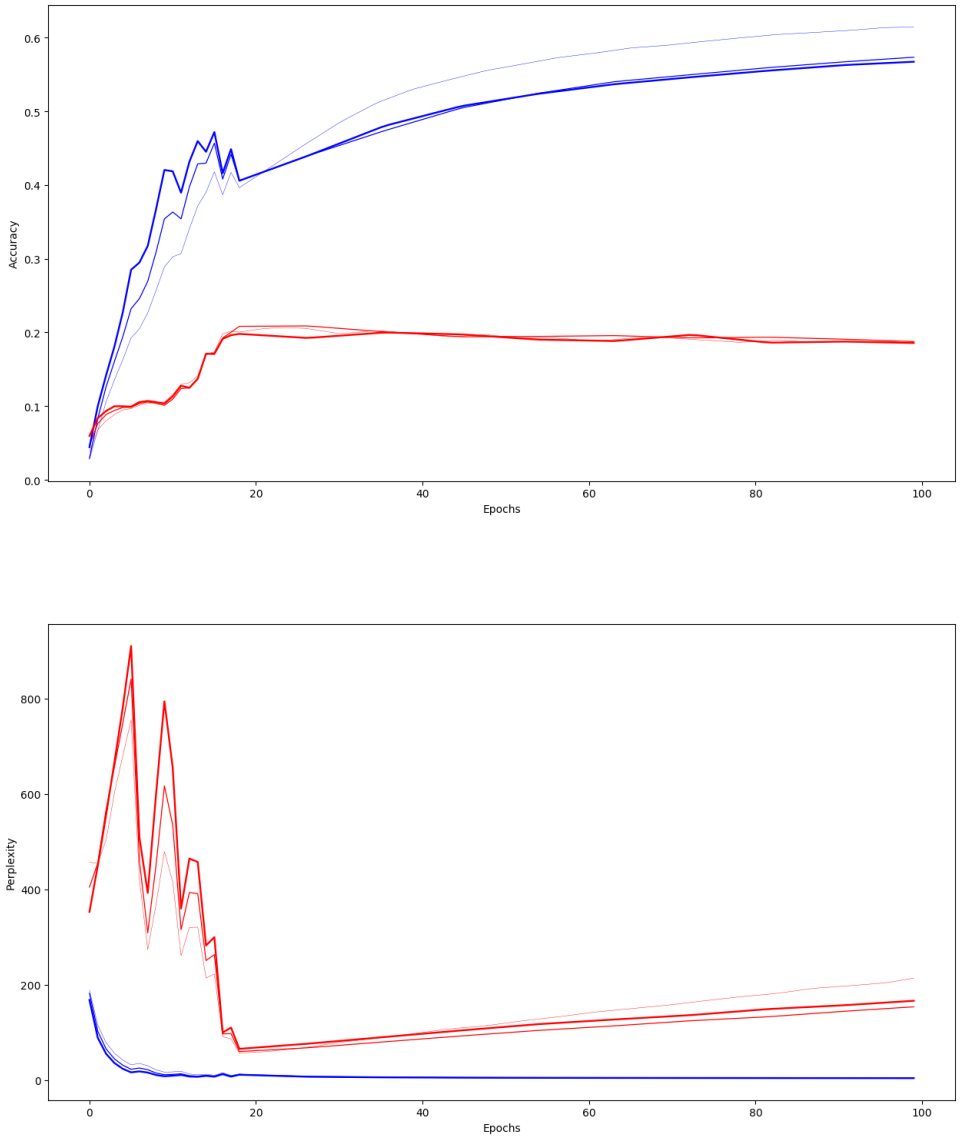

Figure 3: Accuracy and Perplexity comparison for 128(thin), 256(medium) and 512(thick) hidden units

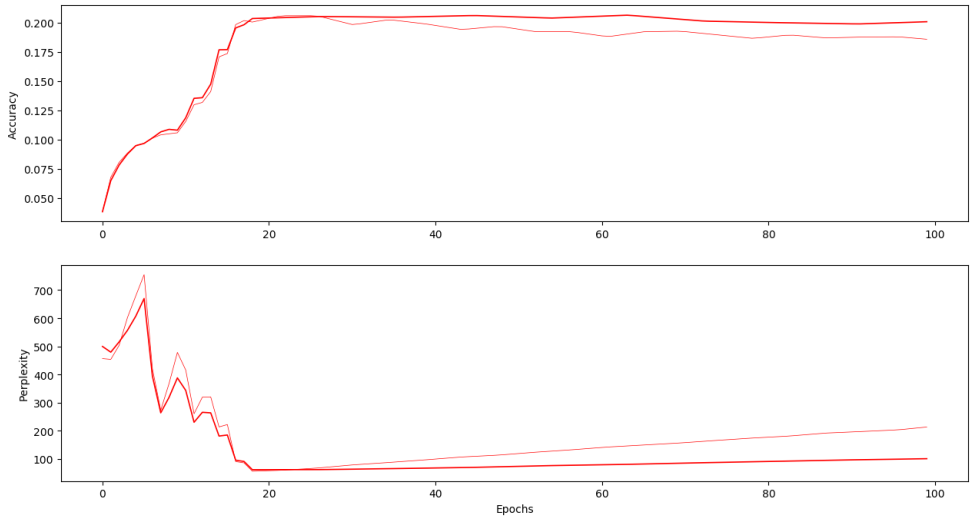

Figure 4: Linear(thin) vs Tanh(thick) activation comparison

- 'financial': 'securities', 'these', 'u.s.', 'york'
- 'chairman': ';', 'group', """, 'department'
- 'bonds': 'stocks', 'funds', 'people', 'companies'
- 'into': 'of', 'by', 'own', 'with'

The word embedding of less frequent words is typically not as informative.

3.5 Embedding Layer Visualization (10 pts)

See Figure 5 and 6
The embeddings below do show groupings for (did, does,do), (japanese,american), (next, recent), (yesterday, since, while, recent) etc. Therefore we find that the 2 dimensional embeddings layer still learns reasonable similarity measures even though predictive power is limited.

3.6 Extra Points (20 pts)

3.6.1 Recurrent Neural Network

See Figure 7
Best performance achieved Perplexity: 48.5 or (267.6 base 2)
Cross Entropy: 5.6
Accuracy: 22.3% (Percentage of 4 grams where the right word was predicted to have the maximum probability)

See Figure 8
RNN slightly outperforms the more basic network. More significantly, it seems to show much lower generalization error.

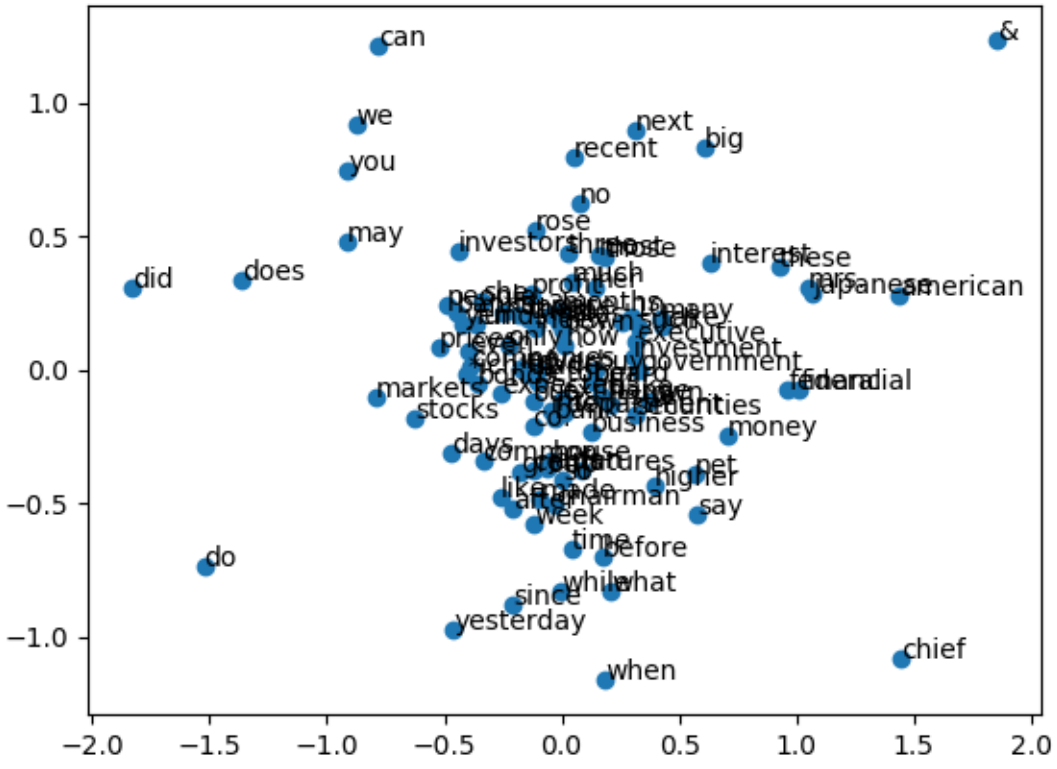

Figure 5: 2D embedding for 100 words

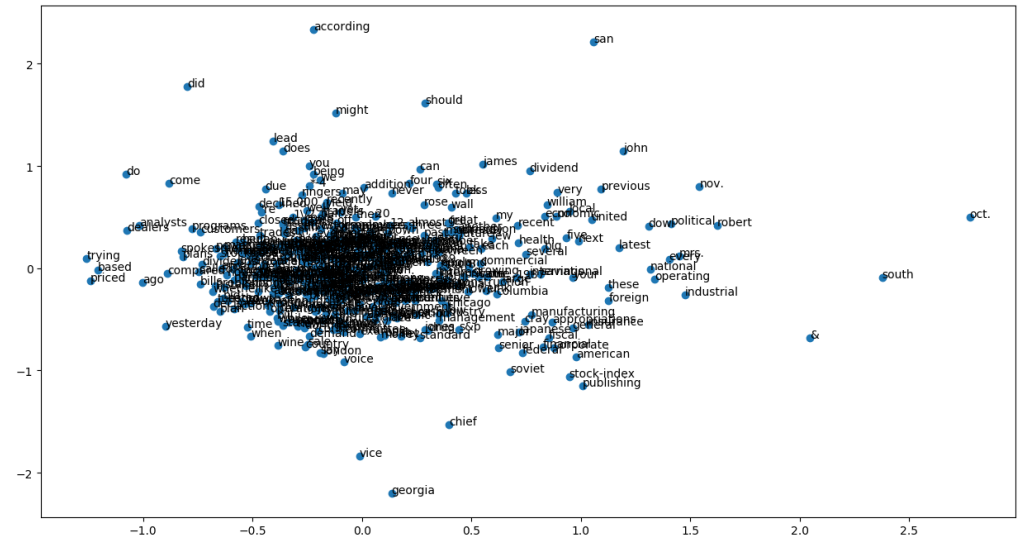

Figure 6: 2D embedding for 500 words

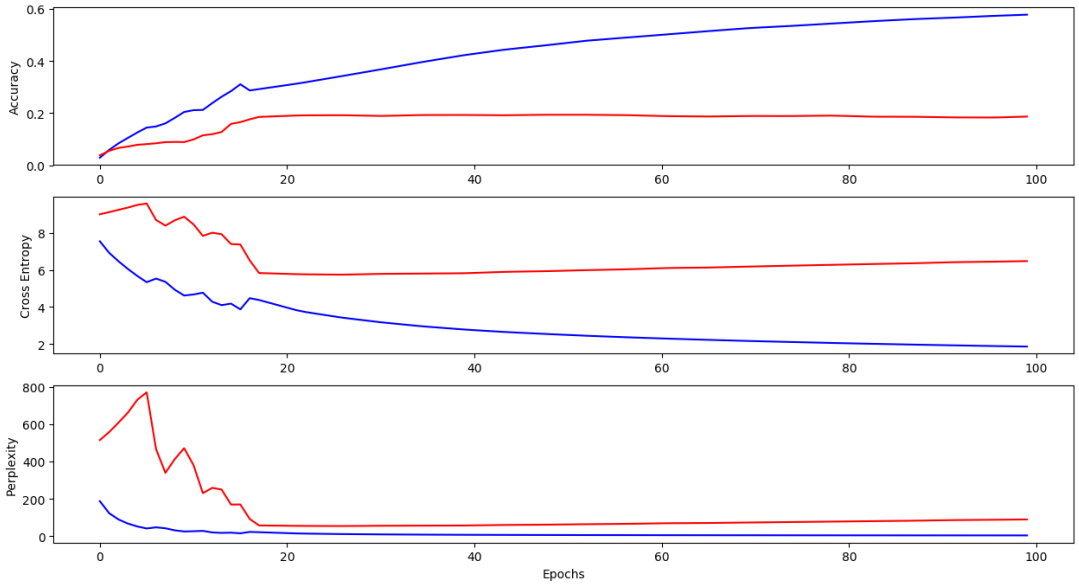

Figure 7: RNN with 16 dimensional embedding layer.

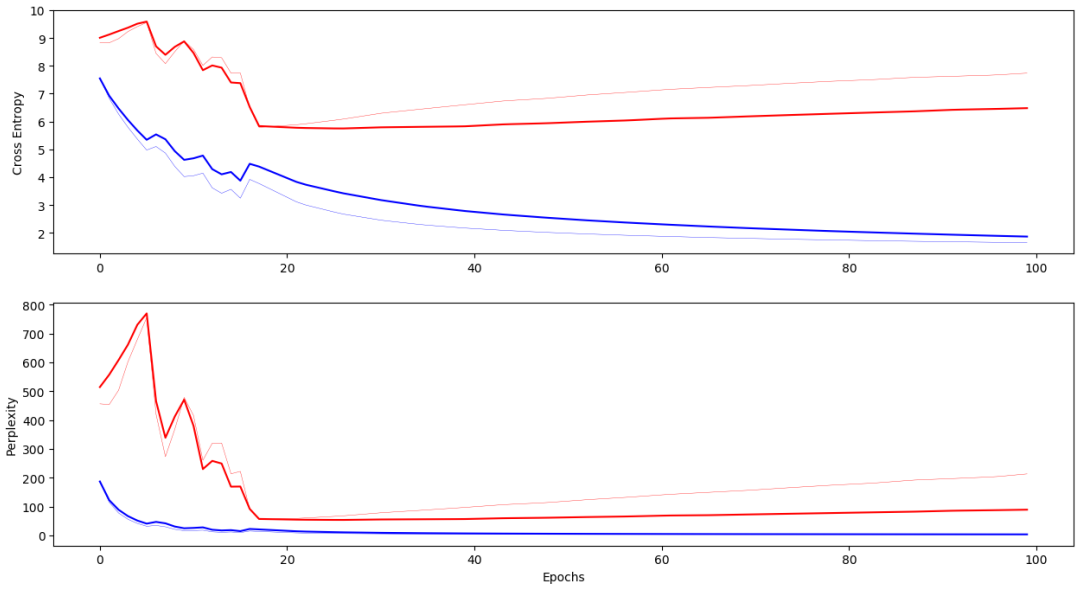

Figure 8: Comparison of basic network in Section 3.2 (thin curve) with RNN (thick curve).

3.6.2 Embedding layer

See Figure 9
The order (Embedding = 32) ¿ (Embedding = 16) ¿ (Embedding = 64) ¿ (Embedding = 128).
The network with 32 dimensional embedding layer outperforms the other, beyond this increasing the model complexity worsens the generalization error.

3.6.3 Truncated Backpropagation

See Figure 10
The truncated network performs worse then the normal network under these hyperparameters. It also has a higher generalization error.

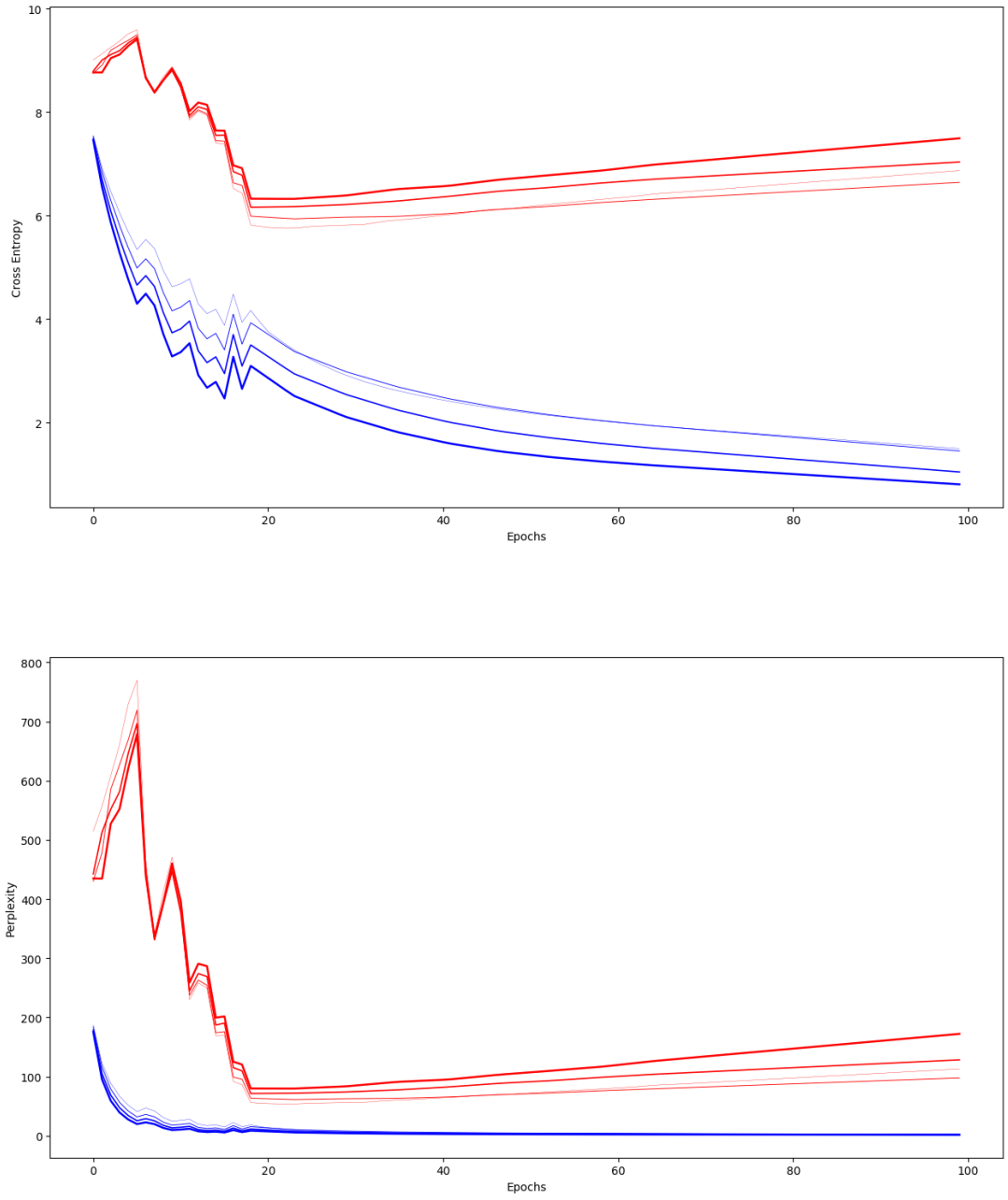

Figure 9: RNN comparison between 16,32,64 and 128 dimensional embedding. From thin to thick curves respectively.

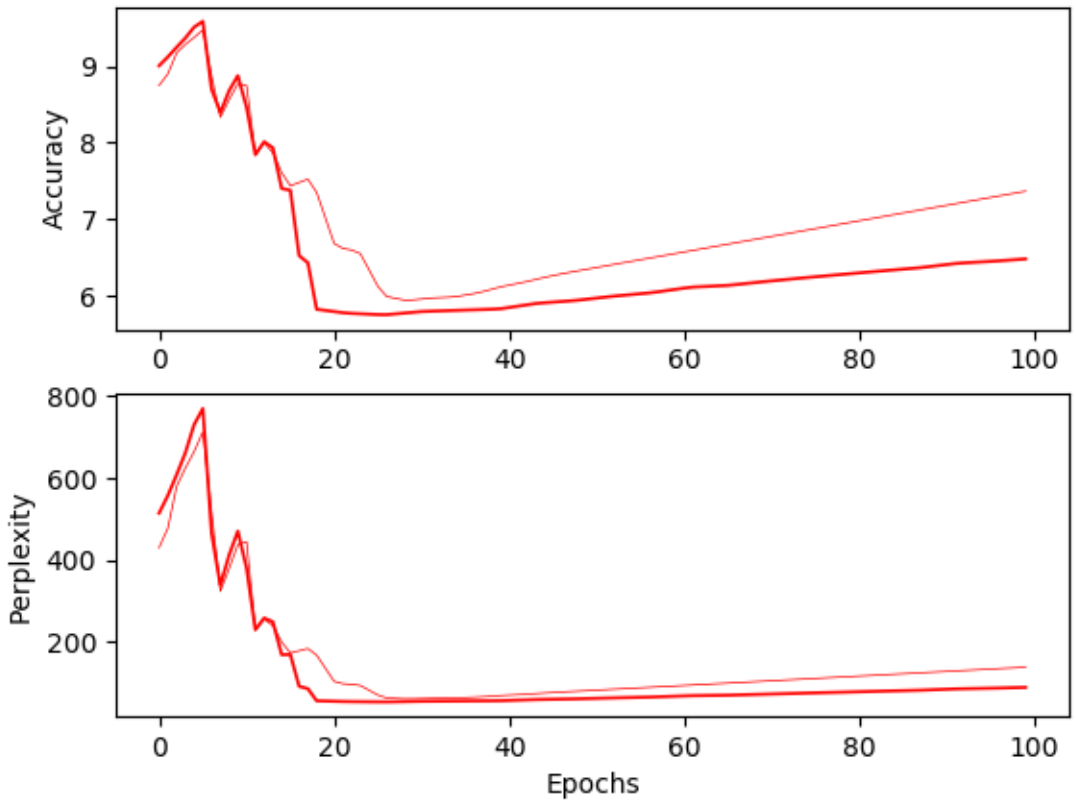

Figure 10: The thin curve indicates the RNN network with truncated back propogation.