# Deep Learning Assignment 2

**Michelle Appel 10170359**
appel.michelle@gmail.com

## 1 Vanilla RNN versus LSTM

### 1.1 Toy Problem: Palindrome Numbers

### 1.2 Vanilla RNN in PyTorch

**Question 1.1**

1. $\frac{\partial \mathcal{L}^{(T)}}{\partial W_{ph}} = \frac{\partial \mathcal{L}^{(T)}}{\partial p^{(T)}} \frac{\partial p^{(T)}}{\partial W_{ph}}$

   $\frac{\partial \mathcal{L}^{(T)}}{\partial p^{(T)}} = -\frac{y}{\hat{y}}$

   $\frac{\partial p^{(T)}}{\partial W_{ph}} = h^{(T)}$

   $\frac{\partial \mathcal{L}^{(T)}}{\partial W_{ph}} = -\frac{y}{\hat{y}} h^{(T)}$

2. $\frac{\partial \mathcal{L}^{(T)}}{\partial W_{hh}} = \frac{\partial \mathcal{L}^{(T)}}{\partial p^{(T)}} \prod_{t=0}^{T} \frac{\partial p^{(t)}}{\partial h_t} \frac{\partial h^{(t)}}{\partial W_{hh}}$

   $\frac{\partial \mathcal{L}^{(T)}}{\partial p^{(T)}} = -\frac{y}{\hat{y}}$

   $\frac{\partial p^{(t)}}{\partial h_t} = W_{hh}$

   $\frac{\partial h_t}{\partial W_{hh}} = h^{(t-1)}(1 - (h^{(t)})^2)$

   $\frac{\partial \mathcal{L}^{(T)}}{\partial W_{hh}} = -\frac{y}{\hat{y}} \prod_{t=0}^{T} W_{hh} h^{(t-1)}(1 - (h^{(t)})^2)$

The gradient with respect to the hidden-output weights is dependent on the current time step only, whereas the gradient with respect to the hidden-hidden weights is dependent on all previous time steps. The latter may result in vanishing gradients for a large sequence, as they multiply.

**Question 1.2**

The vanilla RNN is implemented in `part1/vanilla_rnn.py`. The weights are initialized as random draws from the normal distribution and biases are set to zero. For the forward pass the initial hidden state $h^{(0)}$ is a vector of zeros. For each time step is looped through the sequence of the palindrome. Equations (1) and (2) from `assignment_2.pdf` are used to calculate hidden states and outputs.

**Question 1.3**

As shown in figure 1, the shorter the sequence, the better the performance of the RNN. Using a sequence length of 3 results in perfect accuracy of 1.0, as this is an easy task since the network only has to bridge one time step. Increased sequence length results in reduced performance. It seems that for sequence length 9 and up the network suffers under vanishing gradients and is unable to remember the first digit of the sequence.

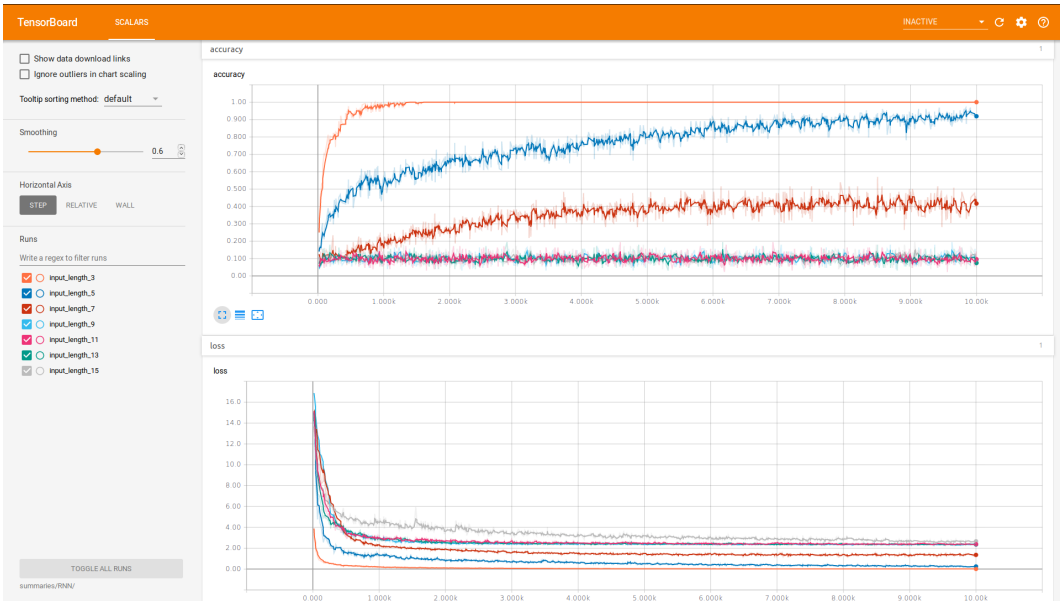


Figure 1: Tensorboard screenshot showing accuracy and loss over iterations for various sequence lengths of the palindrome numbers.

## Question 1.4

In comparison to vanilla stochastic gradient descent, RMSProp and Adam converge faster and obtain better local minima. This is thanks to the use of momentum, which softens oscillations and amplifies speed in the direction towards the optimal.

## Question 1.5

(a) 1. Input modulation gate $g^{(t)}$ uses tanh activation to modulate the input to values between -1 and 1.

2. Input gate $i^{(t)}$ uses $\sigma$ activation to transform the input and hidden state to a value between 0 and 1. The elementwise multiplication between $g^{(t)}$ and $i^{(t)}$ gives the left part of the cell state $(c^{(t-1)})$ (equation (8)).

3. Forget gate $f^{(t)}$ uses $\sigma$ activation to transform the input and hidden state to a value between 0 and 1, where everything that is forgotten has a value near 0, resulting in small values for unimportant information of the previous cell state $(c^{(t-1)})$ in the right side of the cell state $(c^{(t)})$ (equation (8)).

4. Output gate $o^{(t)}$ uses $\sigma$ activation to transform the input and hidden state to a value between 0 and 1.

(b) The total number of trainable features $n_f$ is given by $n_f = 2 * 4dn$

- 2 weight matrices per gate: input and hidden
- 4 gates: modulation, input, forget and output
- d number of features
- n units in layer

## Question 1.6

The LSTM is implemented in `part1/lstm.py`. All weight matrices are initialized as random draws from the normal distribution and biases are set to zero. For the forward pass the initial hidden state $h^{(0)}$ and $c^{(0)}$ are a vector of zeros. For each time step is looped through the sequence of the palindrome. Equations (4) to (11) from `assignment_2.pdf` are used to calculate hidden states and outputs.

The results of the experiment are shown in figure 2. It can be seen that the LSTM network performs better on longer sequences than the vanilla rnn does, probably because of the sophisticated structure of the LSTM network and thus its ability to learn to 'remember' the first digit of the palindrome. However, further improvement might be done by using one-hot vectors instead of index numbers as input, as this assumes dependency between class distances.

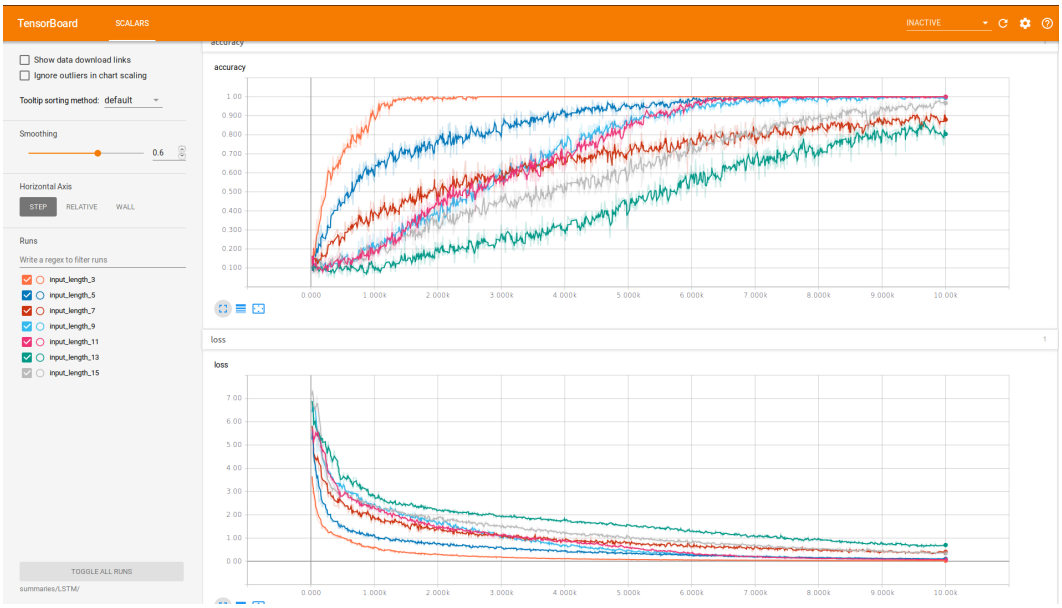


Figure 2: Tensorboard screenshot showing accuracy and loss over iterations for various sequence lengths of the palindrome numbers.

# 2 Modified LSTM Cell

## Question 2.1

## Question 2.2

The gate $k^{(t)}$ controls to what extend the cells are updated normally ($\tilde{c}^{(t)}$ and $\tilde{h}^{(t)}$) and are a duplicate of the previous cell states $c^{(t-1)}$ and $h^{(t-1)}$. When $k^{(t)} = 0$: $c^{(t)} = c^{(t-1)}$, $h^{(t)} = h^{(t-1)}$ and when $k^{(t)} = 1$: $c^{(t)} = \tilde{c}^{(t)}$, $h^{(t)} = \tilde{h}^{(t)}$. The $k^{(t)}$ gate modulates between those extremes as shown in figure 3. This behaviour may be beneficial for data that is periodical, in a sense that some information on some time steps $t$ are more important than others, making $k^{(t)} = 0$ when a time step is not important and $k^{(t)} = 1$ when a time step is important.

## Question 2.3

1. $\tau$ is the size of the period. This parameter can be learned from the data.
2. $r_{on}$ influences the non-zero part size of the period. This parameter can be learned from the data.
3. $s$ is horizontal phase shift. This parameter can be learned from the data.

# 3 Recurrent Nets as Generative Model

## Question 2.1

(a) The two-layer LSTM network is implemented in `part3/model.py` and trained using `part3/train.py`. I chose the book *Alice in Wonderland* to train the network on.

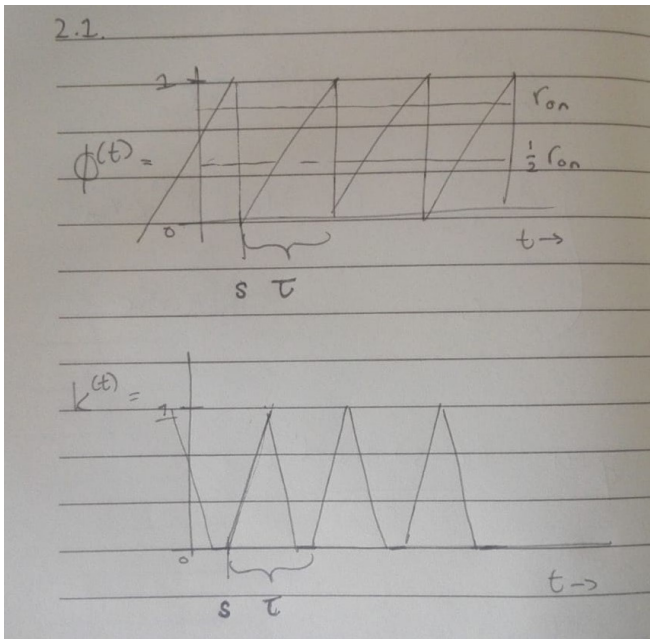


Figure 3: Drawing of how the temporal gate $k^{(t)}$ changes over time.

(b) To generate new sentences `part3/generate.py` is created.

- step 5000: *le the was the was the was the*
- step 10000: *f the same of the same of the*
- step 15000: *Alice was the same thing so sh*
- step 20000: *Hatter was the first thing see*
- step 25000: *I think you'd better not make*

It can be seen that earlier in the training process the network predicts the most occurring words in the book, whereas later on the network had learned to form more sophisticated sentences.

(c) Temperature sampling is implemented in `part3/generate.py`.

- temperature 0.5: *The prisoner's handwriting?' asked another of the jurymen. 'It isn't directed a*
- temperature 1: *Y' said the Dodo, pointing to Alice with one finger; and the whole party at once*
- temperature 2: *Xquirtly's griry. then quietly in melited; aytens:– you'D TM-FOcT'ZCHEREVEE*

As expected do low temperatures approach greedy sampling whereas high temperatures tend to be more random and result in 'experimental' sequences.

**Bonus Question 2.3**

Sampling sentences longer than seq_length is implemented in `part3/generate.py`.

**Experiment I: Whatsapp**

I've trained the network on a Whatsapp chat conversation, which is retrieved by exporting the history file in the chat. The model contains 3 LSTM layers with 256 nodes each and is trained with a sequence length of 120. The generated conversation looks like a drunk text exchange, however an

accuracy around 80% was achieved. This is a sample of the generated output:

10-07-18 10:03 - Pei chu nan: Hahahaha
10-07-18 12:49 - Memechelle: Ik wil dat ook altijd met zeggen
10-07-18 17:05 - Memechelle: Heb je nog wat gezegd om op te vragen dan maakt het niet eens of je me naar recht die is nog iets van de luchtrief is
10-08-18 11:17 - Memechelle: Maar dan moet je er niet op in hahaha
03-08-18 20:33 - Memechelle: Hahahahahaha
03-08-18 13:57 - Pei chu nan: Hahahaha
03-08-18 19:54 - Pei chu nan: <Media weggelaten>
03-08-18 16:17 - Memechelle: <Media weggelaten>
03-08-18 16:47 - Pei chu nan: Maar als je er nog niet aan toe voel eten
09-08-18 15:12 - Memechelle: Hoe was het
03-08-18 20:45 - Pei chu nan: Ik heb een of andere kilkoren ging in slaap gevallen van 300 eten
03-08-18 19:53 - Pei chu nan: <Media weggelaten>
03-08-18 16:17 - Memechelle: <Media weggelaten>
03-08-18 16:47 - Pei chu nan: Maar als ik mn timoteche maandag wel n beetje met school doe eerst
03-08-18 20:35 - Memechelle: Maar ik ben ook best leuke stybooon blijft zijn
10-08-18 11:07 - Memechelle: Oja klinkt wel als ik mastager al dat gesprek
10-08-18 11:06 - Pei chu nan: Ja ik wil ook niet meer over gehakbng en n bogsertuin
10-08-18 11:01 - Pei chu nan: Maar ik was echt meganice vinder maar ik was echt n hoopdor gegaan
09-08-18 15:20 - Pei chu nan: Ik heb er ook echt niks te doen
09-08-18 15:18 - Memechelle: Ik heb echt een paar dagen naar huis
01-08-18 14:32 - Memechelle: Maar dat is ook wel heel erg veel
03-08-18 22:33 - Pei chu nan: Nooooo
03-08-18 16:18 - Memechelle: <Media weggelaten>
03-08-18 16:47 - Memechelle: Ja is ook gaar
01-08-18 12:17 - Memechelle: Hahahahahahaha
03-08-18 13:47 - Pei chu nan: Maar ik weet niet wat ik moet doen

Also, I hoped that the LSTM would learn a logical order of date and time, which it did to some extent, but not entirely.

**Experiment II: Combining 2 books**

I combined Alice in Wonderland and Karl Marx' Manifesto of the Communist Party together in one txt file and trained a network with 2 LSTM layers, 128 nodes each. This is a sample of the result:

I'll tell you my history, and you'll understand, so the Dormouse said–' the Hatter went on, 'and most things twinkled after that–only the March Hare said–' 'I didn't!' the March Hare interrupted in a great hurry, muttering to himself as he came, 'Oh! the Duchess, the Duchess! Oh! won't she be savage if I've kept her waiting!' Alice felt so desperate that she was ready to ask antagonistic to the progress of industry, and to barter truth, love, and honour for traffic in wool, beetroot-sugar, the March Hare will be much the most interesting, and pase with the desire of abolishing the right of personal and nabolition as mouth, and again, and that's all the first figure,' said the Mock Turtle. 'Hold your tongue!' added the Gryphon, and the King said to Alice, the class that holds the future in its hands. 'Ourn them on the other by this tonguered it had made the words all coming different, and then the disappearance of class culture is to him identical with the disappearance of class culture is to him to be afford to geven she liked at the ordarc, streatid. 'What would be grand, certainly,' said Alice, thought to herself. 'I dare say you're wondering why I don't put my arm round your waist,' the Duchess said after a cattoing on the sneeze of the bourgeois relations; to cruses–' mustronmiar, mould the anything in common with the bourgeoisie was accompanied by a corresponding political advance of that class. An oppressed class of beoked on of Agesing on its conditions of existence of his own class. He becomes a pauper, and pauperism develops more rapidly than population and wearts. The Duchess took no notice of them head. 'I have she had put on one of the Mock Turtle. 'She can't explain it,' said the March Hare. 'He turneting his arms and frowning at the cook till hier, is many the mory ard manished in the adminisal geassed to come off a minahing Enour instence, and the political constitution adapted to it, and by the economical and political sway.