1 Theoretical Exercise

1.1 Task 1

The number of parameters can be computed with the following formula : $n^2 + kn + nm$, where n is the dimension of the hidden layer, k is the dimension of the output layer and m is the dimension of the input layer. Using this formula, the number of parameters is 20224.

Since the number of parameters is shared by all the steps of the RNN, the number of parameters stays the same.

1.2 Task 2

1.2.1 Question 1

The problem is with vanishing gradients; to model long term dependencies the back propagation step in the training has to go through many time steps to adjust the weights at the initial time steps. Since, according to the chain rule, the back propagation involves multiplying many gradients which can lead to a gradient, which can tend to zero. A near zero gradient, however is very suboptimal for the gradient step.

1.2.2 Question 2

Bi-directional Can lable elements of a sequence based on past and future elements by processing the input in both directions.

LSTM & GRU Can avoid the vanishing gradiend problem using gates. See next task.

Second order RNN Use higher order weights ($w_[ijk]$ instead of $w_[ij]$) to allow a direct mapping to a finite state machine concerning training, stability and reprsentation.

1.3 LSTM Task 1

1.3.1 Question 1

Gates are a way to optionally let information through; by allowing to let this happen LSTM combat the problem that RNN encountered when modelling long term dependencies.

1.3.2 Question 2

To update the old cell state to the new state following transition is computed :

$C_t = f_t * C_{t-1} + i_t * \widetilde{C_t}$. If the forgot gate is zero, only the new candidate values are considered by the weight i_t

1.3.3 Question 3

LSTM solves the vanishing gradient problem through the gating mechanism; it allows the information to directly pass through (mathematically this means applying an identity function over the inputs and because the gradient of the identity function is 1, the problem of vanishing is reduced by a huge margin)

Another way to look at it is :
n the recurrency of the LSTM the activation function is the identity function with a derivative of 1.0. So, the backpropagated gradient neither vanishes or explodes when passing through, but remains constant. The effective weight of the recurrency is equal to the forget gate activation. So, if the forget gate is on (activation close to 1.0), then the gradient does not vanish. Since the forget gate activation is never greater than 1.0, the gradient can't explode either. So that's why LSTM is so good at learning long range dependencies.

1.3.4 Question 4

A standard LSTM has 3 gates; namely input,forget and output gate which introduce a lot of parameters and make training quite a hassle. One way to reduce the parameters in the LSTM is by simplified gating which was introduced by GRUs. Instead of having two gates, the forget and input gates are combined into a update gate.

1.3.5 Question 5

In a normal LSTM (or generally RNN) we have the limitation, that future information can not be accessed from the current state (because the input is read as a sequence). Bidirectional LSTMs process the input in both directions. This is done by connecting two hidden layers of opposite directions to the same output. This way the output layer can get information from the past and future of the sequence.

As I understand it bidirectional RNN are limited to input sequences of finite length (because we cant start at the end of the sequence if it is infinite).