Lab 7 Exercise - Transforming Sequences

1.1 Complete and train a sequence-to-sequence model

Below is the snippet of code I wrote. It simply consists of an embedding layer and an RNN. The encoder function projects a sequence into a latent semantic space. The following are the same loss curve during training which was nearly continuously falling until it converged.

```
def forward(self, src):
        # TODO
        #All the encoder needs to do is pass the
        #The forward method should return the hidden
        #The output of the LSTM should be ignored
        embedded = self.embedding(src)
        output, (hidden, cell) = self.rnn(embedded)
        return (hidden, cell)
```

Figure 1: The snippet of code

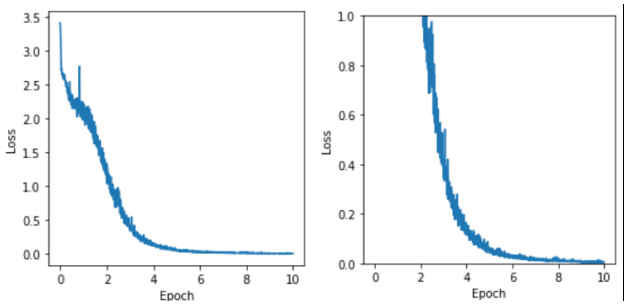

Figure 2: The loss curve during training

1.2 Now use it!

1.2.1 Why is the order of the output reversed?

This is simply because the sequence-to-sequence model output returns a sequence in a reversed way. We need to reverse it back. But I don't know why and how the model reverse a sequence even though I have studied the code of that quite a lot of time.

1.2.2 Teacher forcing:

It is an RNN training trick that replaces the actual output y(t) of a unit by the teacher signal d(t)—the actual or expected output from the training dataset if anyone exists. Teacher forcing allows parameters of an RNN to descend more along their gradients each epoch so that addresses two notorious difficulties in training RNNs: slow convergence and instability.

1.3 Sequence Length

A long chunk can be translated well into a single letter no matter how long it is. In training set, there is a one-to-one mapping from every letter and additional character to Morse code. E.g. 'h' in the training set corresponds '….', and if I pass '……' or '…. –' in decoder, it will still turn out 'h'.

What is more, if I pass a short code into the model, without chunking and with max length equal to the number of letters hidden in the code, it works to decode them out. Yet, if the code is longer, the model fails, for example:

".-- .... -.--" → 'why'

".-- .... -.-- / .. ... / - .... . / --- .-. -... . .-. /" → '^^^^h the eoder '
but it should be *'why is the order '*

This phenomenon may result from the mechanism in LSTM. A LSTM model is trained to have a fixed term of memory based on the given training set. If it then receives a piece of far longer code without chunking at a time, it will necessarily fail to keep that long memory, thereby getting into a mess.