Automatic Speech Recognition

Assignment 3

Maitrey Gramopadhye 160050049
Gurparkash Singh 160050112
Drumil Trivedi 170020016

1 RNN Transducer

1.1 Introduction

The main task of an RNN Transducer is to learn a generalized and robust mapping from one sequence-space to another. This is a very powerful idea since a large number of tasks can be described in this generalized framework, for eg. Language Translation, Text to Speech or, as in our case, Speech Recognition. RNNs have been very successful in various sequence-to-sequence tasks, however, their application is only limited to the tasks for which the alignment between the input and the output is known in advance. RNNs were combined with the Connectionist Temporal Classification (CTC) Loss to alleviate the problem of alignment. The RNN Transducer was proposed as an extension to the CTC approach for sequence labeling tasks in which the alignment between the input and the output was unknown. The RNN Transducer improves upon the CTC Loss approach in two main ways, first, by relaxing the frame independence assumptions, and second, by making it possible for the output sequence to be longer than the input sequence, which was a critical drawback of the previous method.

1.2 Working

Similar to the CTC method, we augment the output space by allowing it to output null character, which intuitively means an empty output. The new space, $\overline{Y} = Y \cup \phi$. Therefore, we have $\overline{Y}$ as the output space and X as the input space (at each time step). Given an input $x \in X^*$ and an output $y \in Y^*$, we define $Pr(y|x) = \Sigma_{a \in B^{-1}(y)} Pr(a|x)$, where B is a function that remove the null symbols from the alignments in $\overline{Y}^*$.

We use two RNNs to compute $Pr(a|x)$. One network, referred to as the transcription network F scans the input sequence x (of length T), and outputs the sequence $f = (f_1, f_2, ..., f_T)$. The other network, referred to as the prediction network G scans the output sequence y (of length U), and outputs the sequence $g = (g_0, g_1, ..., g_U)$. The basic layout of the network is given below.

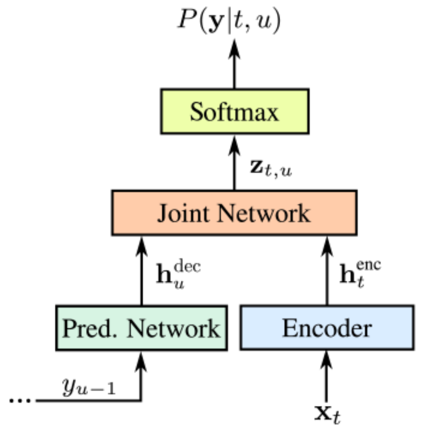

Figure 1: RNN-Transducer

1.3 Prediction Network

For this component, we can use any of the popularly used sequence-to-sequence networks. In the paper, the authors have discussed RNNs and LSTMs. The network maps the $U + 1$ length input $(\phi, y_1, y_2, ..., y_U)$ to a sequence of an equal length on the same output space. Both the input and output are encoded as k-length one-hot vectors (where k is the number of output labels, ϕ is encoded as all zeros). The prediction network attempts to model each element of y, given the previous ones.

1.4 Transcription Network

For this component, the authors have used bidirectional RNNs (or LSTMs). The network maps the T length input $(x_1, x_2, ..., x_T)$ to a sequence of an equal length on the output space $\overline{Y}^*$. The transcription network is similar to the CTC augmented RNN which also uses a null output to define a distribution over input-output alignments.

1.5 Output Distribution

Once we have the transcription vector f_t and the prediction vector g_u, we can define the notion of *output density function*, for a given label $k \in \overline{Y}$ as $h(k, t, u) = exp(f_t^k + g_u^k)$, where the superscript k is used for indexing. This density can be normalised to calculate the *conditional* output distribution:

$$Pr(k|t, u) = \frac{h(k, t, u)}{\Sigma_{k' \in \overline{Y}} h(k', t, u)}$$

Using the above definition, we can define 2 more terms:

$$y(t, u) = Pr(y_{u+1}|t, u)$$

$$\phi(t, u) = Pr(\phi|t, u)$$

Using the above 2 notations, we can construct an Output Probability Lattice (as given below). The lattice can be thought of as an HMM where the node (t,u) represents the probability of having output the first u elements of the output sequence by the point t in the transcription sequence. From each node, the vertical arrow represents the probability of outputting the element u+1 and the horizontal arrow represents the probability of outputting nothing at (t,u). Therefore, the weights of horizontal arrows contribute towards transition and those at vertical arrows contribute towards emission. Using the Forward-Backward Algorithm, we can compute $Pr(a|x)$.

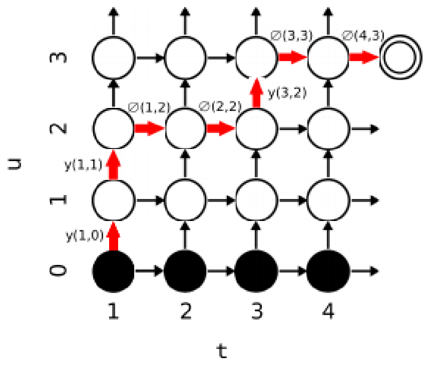

Figure 2: Output Probability Lattice

1.6 Testing

During test time, we use a fixed-width beam search through the tree of output sequences. Even though the authors have implemented the Beam Search, it can be easily extended to an N-Best Search as well, by simply expanding on a sorted list of the N best elements in B instead of just a single best element as done in Beam Search.

2 Relaxing CTC's Frame Independence Assumption

CTC assumes that outputs at each time-step are conditionally independent given the input, i.e the output probability distribution can be computed as -

$$Pr(a|x) = \Pi_{t=1}^{T} Pr(a_t|x)$$

The RNN-T removes the conditional independence assumption in CTC by introducing a **prediction network**, an RNN that is explicitly conditioned on the history of previous non-blank targets predicted by the model. Specifically, the prediction network receives as input the last non-blank label, y_{u-1}, to produce as output which along with the output of the encoder network (Transcription network) is fed to the joint network.

Thus the output probability distribution can be thought of as -

$$Pr(a|x) = \Pi_{t=1}^{T} Pr(a_t|x, a_0, ..., a_{t-1})$$

3 Streaming with RNN Transducer

Attention-based encoder-decoder models cannot be deployed in applications where streaming recognition is required, since they must examine the entire input sequence before they can output any labels. They encode the input sequence by reducing the time resolution and feed this encoded sequence to the decoder (in attention based models this input is accompanied by attention weights) which outputs the probability distribution over output symbols.

Streaming is possible with RNN-T model if a **unidirectional encoder** is used. As, inference in the RNN transducer is performed in a frame-synchronous manner. The unidirectional encoder depends on only the forward hidden sequence, unlike the bidirectional encoder which uses forward hidden sequence as well as backward hidden sequence to calculate encoding, and thus only depends on the input sequence seen so far, which is the requirement for streaming.

4 Limitations of RNN Transducer

A limitation of RNN-T architecture is that training of the RNN-T system is complicated, in order to get good results. The process requires tracing of the output probability lattice, which could be expensive and is not required in other architectures. Also, obtaining convergence while training is slow, using gradient descent.

References

[1] Alex Graves, Sequence Transduction with Recurrent Neural Networks

[2] Kanishka Rao, Haşim Sak, Rohit Prabhavalkar, Exploring Architectures, Data and Units For Streaming End-to-End Speech Recognition with RNN-Transducer

[3] Rohit Prabhavalkar, Kanishka Rao, Tara N. Sainath, Bo Li, Leif Johnson, Navdeep Jaitly, A Comparison of Sequence-to-Sequence Models for Speech Recognition