Automatic Speech Recognition using Transducers

W266 Natural Language processing

Madhu Hegde

mrhegde@berkeley.edu

GitHub: `https://github.com/MadhuAtBerkeley/W266_Project`

Abstract. In this project, I explored the implementations of end-to-end trained Automatic Speech Recognition (ASR) module. The goal was to enhance the performance of decoder module of of state of the art Conformer ASR system [1] by adding Language Model (LM). However, it was found that training and fine tuning the performance of ASR systems is time consuming and results from existing implementations of Conformer were in-consistent. Hence I decided to build ASR module grounds up based on original Transducer paper and understand the complexities of ASR systems. The GRU RNN was used to implement the Transducer and the performance was verified using Librispeech database. The project offered me an opportunity to learn pytorch and write CUDA optimized code for GPU implementation. The transducer achieved WER of 19.86 and I did not see much difference in WER using beam search.

1 Introduction

Before Deep Learning (DL) the generative models such as HMM/GMM were used for Automatic Speech Recognition (ASR). These conventional systems have multiple modules (below) and it is difficult optimize when the modules are trained individually. End-to-end ASR greatly simplifies the model building, training and inference by converting multi-module architecture into a single deep learning architecture.

Since the introduction of deep learning in ASR, a variety of neural network architectures are being proposed. The Transducer model based on Encoder-Decoder architecture is widely used for SOTA ASR systems. The encoder converts low level speech signals into higher level features and the decoder converts these higher level features into output utterances by specifying a probability distribution over text sequences.

The Transducer ASR is a sequence-to-sequence model proposed by Alex Graves [2] way back in 2012. The CTC (Connecitonist Temporal Classification) was used in ASR systems[3] before Transducers and used CNNs to learn feature representations. The CTC assumed conditional independence of output sequence based only on imputs and required separate languange model for decoding. The CTC model is similar to an acoustic model in a traditional ASR system. The RNN transducer [2] augments the encoder network from the CTC model architecture with a separate decoder/prediction network and the joint network. The dense representation of the current input from the encoder and previous prediction from decoder network are joined to combine the auditory and language model into one end to end trainable system. The RNNs were used for transducers to model the sequences and now self attention is used in the encoder/decoder.

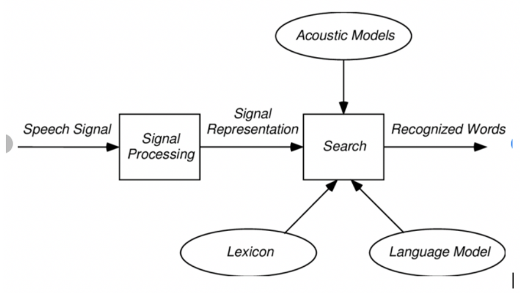

Fig. 1. ASR Model

2 Project Overview

2.1 Dataset Used

Librispeech corpus of approximately 1000 hours of read English speech sampled at 16 kHz, is used as the dataset. The data is derived from read audiobooks from the LibriVox project, and has been carefully segmented and aligned. The table below shows the train/dev/test split in the database.

The dataset is downloaded from www.openslr.org

Table 1. Librispeech Dataset

Subset	Duration	No Speakers
dev-clean	5.4 Hrs	40
test-clean	5.4 Hrs	40
train-clean-100	100 Hrs	251
train-clean-360	360 Hrs	921
train-other-500	500 Hrs	1161

2.2 Feature Extraction

The Mel frequency Cepstrum Coefficients (MFCC) are used as input features. The block diagram of MFCC generation is shown below.

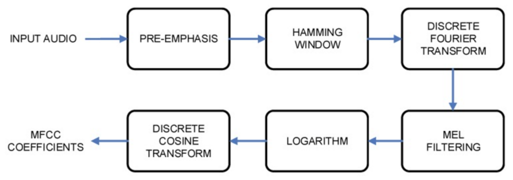

Fig. 2. MFCC Feature extraction

The MFCC values represent human auditory impulse response obtained by homomorphic deconvolution of time domain speech samples. The Hanning window has window size of 20 msec and overlap of 10 msec is used to prevent blocking effect. The DFT size is 512 and 64 MFCC values are used as feature vector. The speech signal can considered to be stationary for 20 msec and each feature vector corresponds to a phoneme unit.

There are many tool-kits are available for feature extraction and I used the Nvidia implementation using Librosa.

2.3 Problem Definition & Approach

The speech recognition is auditory sequence to text sequence generation problem where input and output sequences have different lengths. The ASR is challenging due to following reasons.

- Human speech has huge variation depending on the speaker that makes auditory modelling difficult.
- The word boundaries are not fixed and alignment between input and output sequence is very fluid.
- Some words sound similar (homophones) but can have different meaning/spelling depending on the context

Traditional ASR systems used HMM/GMM for auditory model, dynamic programming/viterbi for phoneme alignment and separate language model (LM) for correct spelling. The transducer ASR models enabled end-to-end ASR by using RNNs for variable auditory sequence detection and encoder-decoder architecture for language modeling. The probability of output sequence is conditionally independent on both the input sequence and past output sequence (unlike older models that were only dependent on inputs).

Given a sequence of input feature vector $\mathbf{x} = (x_1, x_2, .., x_T)$ and output sequence $\mathbf{y} = (y_1, y_2, .., y_T)$, the CTC ASR maximized the conditional probability as given below.

$$P(y/x) = \sum_{\hat{y} \in B(y,x)} \prod_{t=1}^{T} P(\hat{y}_t | x)$$

where $B(y, x)$ corresponds to frame level alignment between input sequence $\mathbf{x}$ (of of length T) and output $\mathbf{y}$ (of length L).

The transducer concept improved the discriminative model by defining $P(y/x)$ as the sum of the probabilities of all possible alignments between

$$P(y/x) = \sum_{\hat{y} \in B^{-1}(y)} P(\hat{y}|x) = \sum \prod_{l=1}^{L} P(\hat{y}_l / (\hat{y}_{t-1}, ..\hat{y}_1, x)$$

Where $\hat{\mathbf{y}}$ has extra null symbol in the vocabulary for alignment it is a set of all sequences with $\mathbf{y}$ and null symbol.

The block diagram of transducer model is shown below.

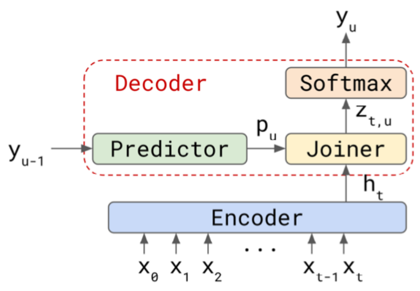

Fig. 3. Transducer Module

The encoder encodes the input acoustic sequence $\mathbf{x} = (x_1, x_2, .., x_T)$ to $\mathbf{h} = (h_1, h_2, .., h_T)$ with potential subsampling $T_s \leq T$. And the decoder contains a predictor to encode the previous non-blank output symbol y_{u-1} for the logits $z_{t,u}$ to condition on. From the illustration, we see that transducer incorporates a language model of output symbols internally in the decoder.

Given a (**x**,**y**) pair, the transducer defines set of monotonic alignments between **x** and **y**. Different alignments are made possible by inserting a blank symbols (ø) in **y**. Since there is no perfect alignment to use as a reference or target, the Transducer defines loss function as

$$-log(P(y/x)) = -log(\sum \prod_{l=1}^{L} P(\hat{y}_l/(\hat{y}_{t-1}, ..\hat{y}_1, x))$$

and the model is trained minimizing this loss function. There are too many alignments possible but fortunately the monotonic alignment constraint imposes a structure where the loss can be computed recursively as shown in the computational graph shown below.

The monotonic alignment is the inductive bias of transducer model and it is justified because the order of characters/words in the output sequence always follows that in the input speech.

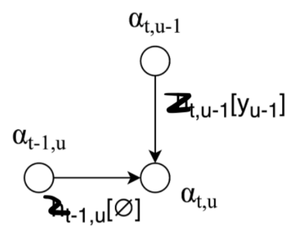

Fig. 4. Forward Loss

The forward variable $\alpha_{t,u}$ for $1 \leq t \leq T$ and $1 \leq u \leq U$ is given as

$$\alpha_{(t,u)} = \alpha_{(t-1,u)} * z_{(t-1,u)}[\text{ø}] + \alpha_{(t,u-1)} * z_{(t,u-1)}[y_{(u-1)}]$$

After computing $\alpha_{t,u}$ for every node in the alignment graph, using the forward variable at the last node of the graph, the $p(y/x)$ is given by

$$p(y/x) = \alpha_{(t,u)} * z_{(T,U)}[\text{ø}]$$

The example of alignments if given below. The two possible alignments for the word CAT is shown.

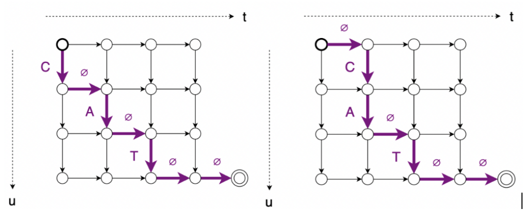

Fig. 5. Alignment

The transducer implementation requires large number of memory - the tensor size = $L*U*B*V$ for batch size of B and vocabulary V. We can see that it easily exceeds 10s of GB for even modest values of L, U, B, V.

The computational complexity can be further reduced by using CNNs to subsample the input features and attention mechanism to avoid $(L*U)$ memory explosion. I plan to improve upon the ASR module with CNN+Attention mechanism in Capstone project.

2.4 Modified Project Scope

Given the large training time (1-2 weeks) of E2E ASR models, I decided to change the scope of the project to build a transducer model that addresses the alignment issue (between input and out sequences) and language modeling. So, the speech signal to text conversion is changed to text to text problem where the input text has repeated characters to mimic the phonemes and some characters changed randomly to represent the auditory channel.

The letters 'g', 'p' and 'j' were randomly removed in the input sequences to simulate auditory model and the insertion of correct letter worked for some words.

The sentences from Librispeech text corpus is modified and examples are given below.

Input: 'wwelll ppriinccee sso eenoa aandd luccccaa aaree nnow uusstt ffamiily eestaatees off the'

Target: 'well prince so genoa and lucca are now just family estates of the'

This simplification reduced the memory and computational requirements by order 100 and it was possible to train within reasonable time.

3 Model

3.1 Baseline Model

The initial plan was to start with open source implementation of Conformer transducer model (uses CNN Transformers) as baseline and improve upon it. So, I started with Conformer model at TensorflowASR. However, the implementation is work in progress with many bugs. It took many weeks to train due to sub-optimal GPU implementation (did not use CuDNNLSTM) and 4 of the issues that I found are at - https://github.com/TensorSpeech/TensorFlowASR/issues?q=MadhuAtBerkeley.

Another major issue I faced is the higher memory requirement of streaming data loaders and training time using legacy pytorch implementations are considerably higher especially with larger batch sizes.

The BufferedShuffleDataset in pytorch 1.8 provides efficient implementation of data generators.

For the modified project scope, I used sequence-to-sequence model with LSTM encoder-decoder without joiner as the base model.

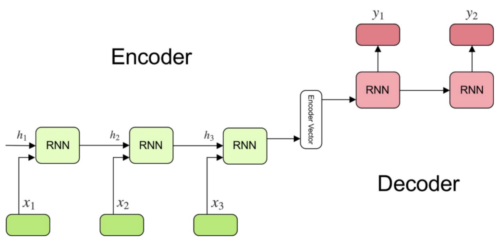

Fig. 6. Base Model: Encoder Decoder

The RNN encoder-decoder is a neural network model that directly computes the conditional probability of the output sequence given the input sequence without assuming a fixed alignment. The encoder maps the input sequence into a fixed-length hidden representation c_o,i.e., context vector. The decoder computes each output value y_o as

$$P(y_1, .., y_o|x_1, , .., x_T) = \prod_{o=1}^{O} P(y_o|y_1, .., y_{o-1}, c_o)$$

The sequence to sequence encoder-decoder model with cross-entropy loss criteria is not expected to work well for speech recognition. This is because there is no one perfect alignment that can be used as reference of perfect alignment.

The base model used cascaded LSTM of 2 stages and 1024 LSTM cells in both encoder and the decoder. Dropout of 0.2 is used both in encoder and decoder. The vocabulary is same as Transducer model with 26 letters , space and single quote. The ¡start¿ and ¡end¿ are added to the input sequences to mark span of the input sequences. The ¡blank¿ (decoder outputs nothing) is added to alignment input and output with different lengths.

The teacher ratio of 0.9 is used during. training - the decoder uses decoded symbol instead of input sequence 10 percent of the times.

3.2 Data Preparation

The computational complexity of transducer depends on the length of the sequence L, U, V. The vocabulary size V is limited to 29 (26 lower case alphabets, blank symbol, single space and single quote). So, the text is converted to lower case with special characters removed. The characters are mapped to integers from 1 to 28 and blank is mapped to 0. The BufferedShuffleDataset combined with IterableDataLoader is used for implementing iterable data loader with configurable buffer size and shuffle option. This is most convenient for handling large dataset spread across multiple files. The buffer size of 1024 sentences is used (trade-off between number of disk accesses and RAM usage).

The maximum length of a sentence in Librispeech is 398 words (24.5 seconds) and it is too long for transducer implementation. The minimum is 8 words(1.4 seconds). The longer sentences are split into smaller sentences with max length of 10 words to reduce memory requirement.

3.3 Transducer Model

Transducer Encoder The bidirectional GRU RNNs are used for sequence modeling. The GRUs have less number of gates and computationally faster compared to LSTMs with comparable performance. The size of hidden layer is 1024 and 3 level cascaded GRUs are used. The embedding layer with vector size 32 is used for input characters. The dropout of 0.1 is used at GRU outputs for generalization.

Transducer Decoder The decoder uses GRUCell RNNs to implement auto-regressive model. The embedding layer with vector size 32 is used for input characters. The decoder uses same embedding (vector of size 32) as the encoder and teacher ratio of 0.9 for training. It means 10% of the time predicted character (instead of target) is used as previous character for during training. The GRUCell of each sequence in the batch is initialized with blank symbol.

During inference, softmax layer is added to the decoder. the greedy search was initially used to find the labels with maximum likelihood. I implemented beam search to see if there is an improvement wrt greedy search. The softmax layer was not required for greedy search but beam search required softmax to generate probabilities that can be used for path metric.

Transducer Joiner The joiner is a simple feed-forward network that combines the encoder vector h_t and predictor vector p_u and outputs a softmax $z_{t,u}$over all the labels, as well blank output ø as shown below.

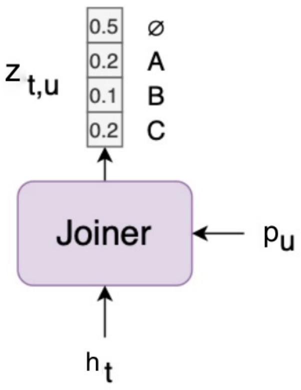

Fig. 7. Joiner

3.4 Model Training

The Adam optimizer is used with exponential learning rate decay. Initial learning rate is 0.0001 and decay rate of 0.9 is used. Given the large variation in the length of input sentences, larger batch sizes are avoided as many smaller sentences would have significant padding. The accumulation of gradients across sentences within a batch is not used as sentence have different lengths.

The length of sequences/sentences is limited to 10 words and batch size of 32 sentences for training and 8 for evaluation. The padding with blank symbols used for padding and match the length of all sentences in a given batch. The max sequence length (10) and batch size (32) are mainly chosen to manage the computational complexity. The performance would be better with longer sequence length (more context) and higher parallelism with larger batch size.

The model is trained for 20 epochs and train data is shuffled to improve training convergence. The eval data is un-shuffled so that decoded sentences are logged at regular interval to check improvement over epochs.

4 Results

In this section, the results of 3 models - open source Conformer model (At TensorflowASR), the Seq2Seq base model and my implementation of Transducer are presented.

4.1 TensorflowASR Conformer

After a week of debugging/fixing the bugs, I could train the Conformer model using 960 hours of LibriSpeech data for 2 weeks (20 epochs and 520k batches).

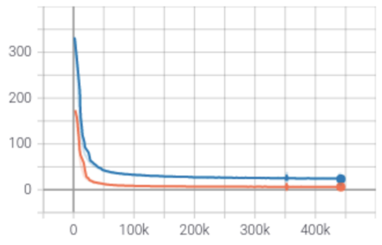

Fig. 8. Conformer Loss

The blue plot is training loss and orange is the validation loss. The training loss shows high bias. The greedy search and beam search showed similar WER values and hence I decided not to pursue building launguage model on top of this Conformer model.

Table 2. WER of Conformer Model

Dataset	WER
Test-clean	6.78
Test-other	15.86

4.2 Base Model

The encoder decoder model with cross-entropy loss did not work well initially with LSTM length of 512. The improvement was dramatic with LSTM length 1024 and teacher forcing ratio of 0.9. The Base model was trained for 40 epochs.

Table 3. WER Base model

Dataset	WER
Test-clean	9.79
Test-other	9.83

4.3 Transducer Model with RNN

The transducer model with RNN-T loss is able to decode/predict most of the words in the test data set. The alignment worked most of the time by removing repeated letters. However, there was an issue in detecting words that have repeated letters as part of the spelling.

The base encoder-decoder model with teacher forcing ratio performed better when it comes to alignment.

I did not find much difference in WER performance between greedy search and beam search. This was surprising and I looked at the path metrics of n-best sequence to understand why. The path metric is generated as sum of logarithm of softmax values at the decoder output. The difference in softmax values of the best symbol and next symbol was so large that the next best path had a huge path metric difference wrt greedy search. This points to suboptimal training where the model has not seen enough training examples to generate softmax values close enough to facilitate other paths/option.

I did not use the length normalized probability (shorter sequences are preferred otherwise) and could not find a way to incorporate the normalization during beam search.

One way to measure effectiveness of training (other than RNN-T loss) is to check how dense are softmax predictions and n-best sequences. If the softmax values are not dense, it implies that training data is not sufficient.

Table 4. WER Transducer model

Dataset	WER(Greedy)	WER(Beam)
Test-clean	19.86	19.96
Test-other	19.20	19.32

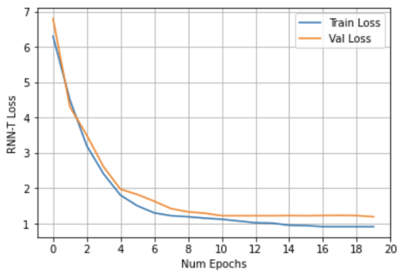

Fig. 9. Transducer Loss

5 Conclusion

In this project it is shown that RNN transducers based on bidirectional GRU can be used for end-to-end speech recognition task using LibriSpeech database. The transducer's ability to align the input and output sequences and insert missing characters (LM) is verified. However, I could not establish RNN-T loss criteria is better than cross-entropy loss when it comes to WER performance.

– The transducer implementation is computationally very intensive as it takes $O(N^2 * V)$ computations where N is sequence length and V is vocabulary size. The inductive bias of monotonic transition of labels helps to simplify the computation of transducer loss.
– End-to-End training with speech signal alone is not enough for language modeling (especially for proper nouns). Other methods such as shallow/deep fusion of external language models at the predictor output can be used.
– Beam search requires optimal training to see performance benefits

Next Steps I would like to continue this work for my Capstone project with following changes.

– Reduce computational complexity by pre-compute/store of MFCC and use real speech samples.
– Bridge WER gap between the base model and Transducer model (Teacher forcing and training)
– Sub-sampling (CNNs) and use Sentencepiece vocabulary.

References

1. Anmol Gulati et al., "Conformer: Convolution-augmented Transformer for Speech Recognition," https://arxiv.org/abs/2005.08100

2. Greaves, A., "Sequence Transduction with Recurrent Neural Networks," 14 Nov 2012 https://arxiv.org/pdf/1211.3711.pdf

3. Graves, A., Fern ande. "Connectionist temporal classification: La- belling unsegmented sequence data with recurrent neural networks" In ICML, 2006

4. Watanabe, S. et al., "Hybrid CTC/Attention Architecture for End-to-End Speech Recognition," TR2017-190 October 2017

5. Rohit Prabhavalkar, "A Comparison of Sequence-to-Sequence Models for Speech Recognition" Interspeech 2017, ISCA (2017)

6. Lu, Liang. et al."A Study of the Recurrent Neural Network Encoder-Decoder for Large Vocabulary Speech Recognition"