Deep Learning Project Report

Speech-to-Text Recognition System

Group Members:

Gayane Vardanyan
Yuhsuan Chen

January 2021

Introduction

Speech recognition systems have become one of the most interesting applications of natural language processing- from Google assistant to Siri, from smart home systems to Cortana, they have become an integrated part of everyday life. Deep learning systems, however, introduced revolutionary capabilities in the field of speech recognition. Exploiting the deep learning advantages of learning from huge datasets, it's possible to create very competitive speech-recognition systems.

Some of the most robust systems in today's market are built due to deep learning, exploiting recurrent neural networks as the base stone for the architecture, for example, Deep Speech or LAS. The first one takes advantage of the CTC loss function in speech transcript prediction, while the latter focuses more on sequence-to-sequence network architecture with predictive aims.

In the frames of this project, we will try to build our own small speech-to-text language recognition system. The main idea of the project is to perform in-depth research on existing state-of-the-art models in speech recognition systems, understand the role of deep learning in building a similar application and explore the capabilities of Pytorch while hands-on building a speech recognition system.

The Dataset

As our training dataset, we use the LibriSpeech dataset. The latter is a large NLP corpus containing about 100 hours of transcribed speech data obtained from English language audiobooks. Torchaudio is used to download the dataset. Each sample from the dataset contains metadata, such as the waveform, sample rate of audio, the utterance/label, etc.

Data Preprocessing

Before working on the actual model, a number of transformations were applied on the dataset, to bring it to a form the model could use. The work with audio-data was majorly simplified with the help of a Pytorch library, called ***Torchaudio***.

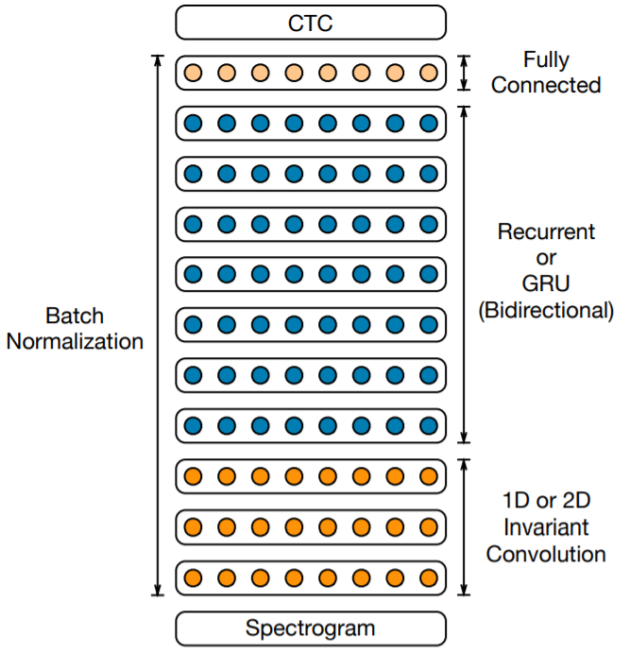
We transformed the waveform audio into a ***MelSpectrogram***, which is basically like a spectrogram (a visual representation) of the sound, and is more convenient for use, compared to the regular Hz representation for frequency. We applied the non-linear transformation to map frequency into Mel scale to represent the y-axis frequency. It can more accurately represent what the human ear heard, and luckily, Torchaudio supports several kinds of transformation, and MelSpectrogram is one of them.

Then, since the dataset we were using was limited, we came across the need of diversification of the dataset with the help of data augmentation. That's when we took advantage of the SpecAugment method, which is a simple and efficient data augmentation method. The essence of the method is that in order to increase the generalization ability of the model, it's often enough to cut out consecutive blocks of time and frequency dimensions. More details about the method

and its applications are present in the above mentioned paper. Most importantly, we used FrequencyMasking and TimeMasking to modify the training Mel spectrogram to add more diversity to the network's input and avoid overfitting the model.
After augmentation, we need to encode the text label into a numerical value. First, we do text cleaning like removing punctuation and multiple spaces, expanding common English contractions, and unifying them into lowercase text. Then we use an integer to represent each character and map each label into a sequence of integers.
After these transformations were applied, we started on the creation of the actual model.

The Model

We build our model majorly following the practices of the Deep Speech model, as described in Deep Speech 2: End-to-End Speech Recognition in English and Mandarin paper. Considering that we're dealing with a very large dataset, a single recurrent layer would not suffice, hence the model capacity is increased via depth, with explorations including many bidirectional recurrent and convolutional layers. ***Batch Normalization for RNNs*** is used for optimization purposes, substantially improving the final error and majorly accelerating the training. The idea is that for scaling purposes we don't make each layer larger, but instead increase the depth of the networks.
The image below demonstrates the architecture of the model, while changing the number of convolutional layers from 1 to 3 and the number of recurrent or GRU layers from 1 to 7:



As for the selection of the recurrent architecture- among a variety of alternatives, the two most commonly used recurrent architectures are the ***Long Short-Term Memory (LSTM)*** and the ***Gated Recurrent Units (GRU)***. In the frames of this project, we decided to examine GRUs, since while similar to LSTM in accuracy, GRU has proven to train faster and the divergence likelihood is comparatively low, as well as can benefit from batch normalization. Moreover, it was

found out that for a fixed number of parameters, GRU architecture achieves better WER. Hence we conclude that the GRU networks outperform the simple RNNs.

Loss Function

Since the model is trained to predict the probability distribution of the alphabet characters per frame in the spectrogram we feed into the model, there might have been a necessity of alignment of the transcript text to the audio. The idea then would have been to train the model to predict specific labels in specific frames. However, instead of that additional pre-training step, we decided to take advantage of the CTC Loss function as it is described in Deep Speech 2: End-to-End Speech Recognition in English and Mandarin paper. Due to the "blank" label introduced by CTC, the model basically learns to align the transcript directly during the training, since the model is now capable of indicating that certain audio frames didn't produce characters. In the source code of the project, we took advantage of the inbuilt PyTorch CTC loss function, however more details on how the function works could be found here.
The CTC loss function is also built into PyTorch.

Evaluation Metrics

There are several metrics that could be used for evaluating the model we created. After consideration we stopped on the two most commonly used, those being WER (Word Error Rate) and CER(Character Error Rate). The idea behind WER is that it compares the output of the model with the original transcription of the speech and computes the overall error based on the number of wrongly predicted words. On the other hand, the CER measures the error based on the number of differing characters. The functions responsible for the computations of the above-mentioned error metrics are called ***wer*** and ***cer*** respectively in the source code provided.
We've used ***Levenstein distance*** for comparing the differences between the words. In essence, the Levenstein distance is the minimum number of single-character edits, namely insertions, deletions, or substitutions, necessary to convert one word into the other.

The results

We evaluated the model against 2 splits of the ***LibriSpeech*** dataset (train-clean-100 28539 samples, dev-clean 2703 sample) across two different epochs to analyze the behavior of the model. The results of the tests are present in the table below. We see from the results below that with a larger training data size and a higher number of iterations, we obtain better results.
We limited our analysis to this configuration owing to the long time taken to train the model with limited hardware.

Epochs	Dataset	Average Loss	Average CER	Average WER
10	dev-clean	1.9498	0.587450	1.0278

20	dev-clean	1.6121	0.577153	1.3062
10	train-clean-100	0.8204	0.252779	0.8374
20	train-clean-100	0.6551	0.196971	0.7548

A sample output provided by the model is given below:

Actual	**Speech Recognised by the Model**
i named nine others and said.	i name nin others and said
two hundred warriors feasted in his hall and followed him to battle.	two hundr waor faton his hall am folle him te battl
i wish you good night she laid her bony hands on the back of mister meadowcroft's invalid chair cut him short in his farewell salutation to me and wheeled him out to his bed as if she were wheeling him out to his grave.	i wish you gold nigt she la her bony hands n the back of mister matacros inbly chaire co him short his far well soliautation the me and weld am out to hs bed as if she wierl walling him out to his grave

We've included the complete results of running experiments in the ***Evaluation Experiments.txt*** file into the project repository.