Introduction :

Speech recognition is an important application used nowadays, and it’s first example was a system designed to recognize 9 digits in single voice in the 1950s. Many researches afterwords were done to improve it to make it applicable to real life applications, so that when a person is talking, a machine is able to recognize what the person is saying which of course is done by machine learning and complicated models of DNNs (deep neural networks), CNNs(convolutional neural networks) and RNNs(Recurrent neural networks). Our work in speech recognition involves using a model that has already been tried before called the Conformer(Convolution augmented transformer) model[1], The dataset used in the previous trial is Librispeech dataset.

The usual pattern in speech recognition, machine translation and other sequence to sequence models is that the model consists of an encoder taking in the input and a decoder producing the output and their contents differ from a model to the other. But the decoder usually consists of one or many RNN(LSTMs or GRUs) layers in a sequence. And what our model encoder consists of particularly helps to improve the speech recognition learning task. There is also a language model trained on the librispeech corpus and tokenized with the WPM(Word Piece Model) mentioned in [2], Our model is implemented using keras API in python while in it’s previous trial it was implemented using pytorch, and after we finish implemeting the model on keras we will be implementing a part of it digitally on the FPGA using Vivado.

The model

As mentioned above, the model consists of an encoder, a decoder and a language model along with it’s word piece model, we will study each of them in detail mentioning the part we designed digitally.

1. **The conformer encoder :**

Figure (1) is a flow chart showing the structure of the encoder:

Conformer blocks X N

Dense Layer

Convolution Subsampling

Spec Augment

Figure (1) : The encoder

* 1. Spec Augment :

In this block we apply a mask in both time and frequency domains, and when we say masking we do not mean blocking we mean weighting or filtering.

* 1. Convolution Subsampling :

This block consists of 2 layers of 2D convolutions with 2\*2 filters and a stride equal to 2.

* 1. Dense layer :

It is just one neural network layer with a certain number of units determined during experimentation.

* 1. Conformer block :

Figure (2) is a flow chart showing the structure of the conformer block:

Convolution module

Multihead Self Attention Module

Feed Forward module

Layer Normalization

Feed Forward module

Figure (2) : The Conformer block

As shown in the above figure The conformer block consists of sequence of modules, but these modules are residual which means that the output of each module is added to it’s input before going into the next module, and for the feed forward modules it’s output is multiplied by ½ before adding it to the input.

* + 1. Feed Forward Module:

This module consists of the following sequence of layers:

* + - 1. layer normalization:

We subtract the mean of the whole layer from each element and divide each element by the standard deviation and then we apply a pointwise multiplication between the resulting vector and another vector with trainable weights and add a bias vector also with trainable weights.

* + - 1. Dense layer:

The number of units for it is determined during experimentation.

* + - 1. Swish Activation :

Each element in the layer is multiplied by it’s sigmoid activation.

* + - 1. Drop out:

A kind of regularization to prevent overfitting of the model, it is applied with a certain drop out rate.

* + - 1. Another Dense layer.
      2. Another Drop out.

This module is the one we designed digitally using vivado.

* + 1. Multihead Self Attention Module:

This module consists of the following sequence of layers:

* + - 1. Layer normalization.
      2. Multihead attention:

The Self attention is known to have a single dense layer for each of the query, key and value of the attention mechanism,the Multihead Attention on the other hand has a certain number of dense layers for each of them which is represented by the number of heads which is a parameter we can control while experimenting.

* + - 1. Drop out.
    1. Convolution Module:

Normally the Transformer consists only of the Multihead Self Attention Module along with the FeedForward Module. But while Transformers are good for long range information, Convolution is good for obtaining local information. So the Convolution Module was added to the Transformer and the model was called The Conformer. The Convolution Module consists of the following sequence of layers:

* + - 1. Layer normalization.
      2. Pointwise convolution.
      3. 1D depthwise convolution.
      4. Batch normalization:

The techniques of normalization used here are the same as the layer normalization.

* + - 1. Swish Activation.
      2. PointWise convolution.
      3. Drop out.

1. **The Conformer Decoder:**

The decoder for this model is simple and of consists of the following layers:

* 1. An Embedding layer :

Which has a length equal to the vocab size of the WPM and a width equal to the embddding representation of each word.

* 1. One LSTM layer:

Where the number of units for each timestep is the vocab size.

* 1. Dense layer:

The number of units for the layer is the decoder width.

* 1. Another Dense layer:

The final dense layer and has a number of units equal to the maximum sequence of words the model is supposed to output.

1. **The Language Model and the Word Piece Model:**

First we built the word piece model which as it’s name suggests builds it’s vocab by combining pieces of words from a list of candidate word pieces, and each word or subword in the vocab has an ID equal to it’s index in the vocab and the new word that enters the vocab each time is the word from the candidate word pieces that most occurred in the librispeech transcripts, so we basically count how many times each word appeared in the transcripts.

And after building the vocab, we built a vocab tree where each node in the tree is a character that has other children nodes which are also characters. So while searching for a certain word in the tree we know which character node to go to and the second character will be one of it’s children and so on until we get the whole word or not(If the word doesn’t exist in the vocab then we will have to divide it into subwords that exist in the vocab and the word we are searching for will have many IDs). This method is very useful for obtaining the IDs for the words we are searching for very fast and it has another name which the Trie data structure.

Figure (3) shows an example for a vocab tree:

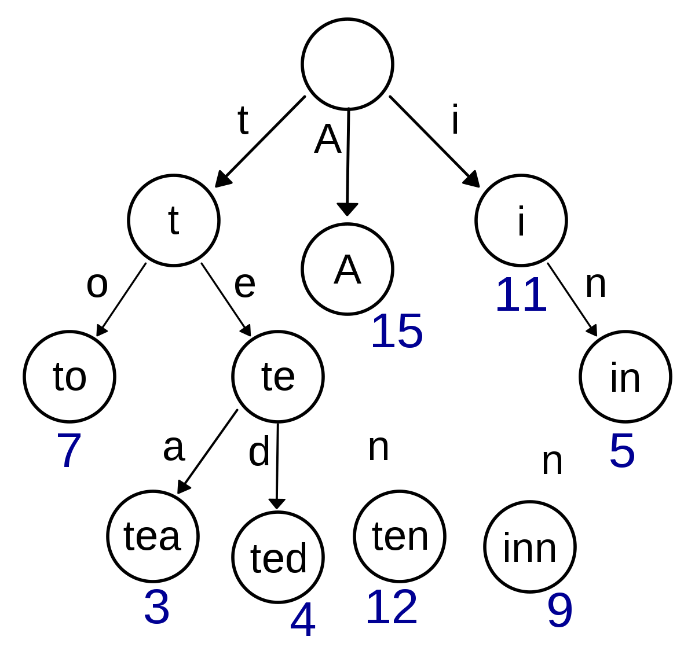


Figure (3) : The Trie data structure

After building the WPM and the tree we built the language model which consists of the following layers :

* 1. An Embedding layer similar to the one in the decoder.
  2. A sequence of 3 LSTM layers:

The 3 layers have a 4096 units in width.

* 1. Dense layer :

The number of units in this layer is the language model width.

* 1. Dense layer :

The final dense layer has a number of units equal to the vocab size because the output is a softmax of probabilities so that the word or the subword with the highest probability is the output.

References :

1-Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, Ruoming Pang, “Conformer: Convolution-augmented Transformer for Speech Recognition,” arXiv:2005.08100, 2020.

2-Mike schuster and and kaisuke nakajima, “Japanese and Korean Voice Search”, 2012.

3-