In The name of God

University Tehran

Engineering Facility

Electrical and Computer Engineering

Deep Learning with Applications

HW # 4

Amin Fadaeinedjad

810195442

Spring 99

Abstract

In this Computer Assignment we are going use the Pytorch framework for the speech to text recognition. We are going to implement the transformer that will do this job for us is to convert a speech signal to a text and for this reason we are going to need lots and lots of data, after that buy implementing that transformer network which contains a embedding part and a encoder it will try to return a sequence of numbers in the output and if use the dictionary from the uploaded files we have two dictionaries called $id2label$ and $label2id$ and this will help us in transferring the numbers from the network to see either the output we have is reasonable or not. In every epoch we also are going to calculate the CER also known as character error rate and the WER or the word error rate and we are going to calculate this parameters as well. In this file we are also going to give a brief over the network and how does this network actually work.

Part 0 – Download and Pre-Process Data

In the beginning of this project we need to prepare the data and because of the large amount of data that can't be done easily at first we need to make a new environment for this thing because there is about 20GB data that need to be downloaded and even google drive doesn't have that much storage and we need to use the local option mode where the it will reset in about every 12 hours.

We need to run the $download\_prepare\_data.sh$ file which was uploaded, this file will to some of the job for us and download and prepare the data for us and that is local folder and we have 3 .csv files which in every one of the locates the location of the audio data and the text of the data, but we have problem with this .csv files they weren't much accurate at all so in this case we cleaned it and got rid on the locations where the text and audio data didn't existed and this was only done by the train data because the other two folders of test and validation dataset had much more problems than the train dataset so I decided to divide the train dataset to 3 datasets such as clean train, test, validation but the way that we are going to use it is quite different we are going to use them in the for loop of the training with this difference that we are not going to update our network with this loss and the data received from it so it might not be a good idea but still because of the not wealthy prepared datasets we must still have validation and train data for our computations. But also in the train dataset there still was a number of data which did not exist so we needed to clean that too so by just using the prepared libraries in python we got rid of the non-existing data locations. We also saved the data as a zip file so without downloading it we can us it easily every time (but to be honest in this case still be got a lots of errors and that was hard for us to wait that long for it so in some cases we just downloaded the data again which we know isn't a good option)

Part 1

In this part we are going to implement the transformer network and see the tasks of the models components.

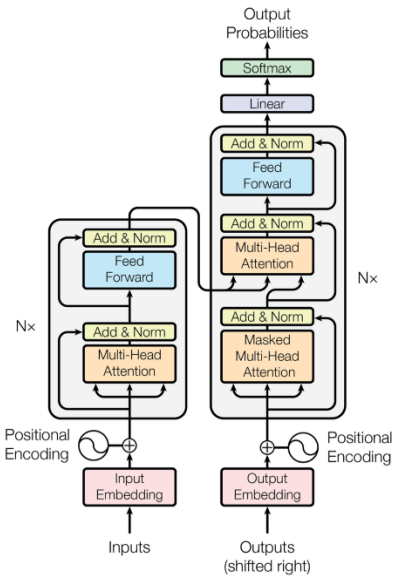

Figure 1: The Transformer - model architecture.

Figure 1 shows the structure of the model that we have mentioned before in this document, the model contains two entrance for the model which one is the $inputs$ and the other is $outputs(shifted\ right)$ which works like a RNN network, and we are going to talk about every component of the network.

Transformer Network Modules:

- Input Embedding
- Output Embedding
- Position Encoding
- Encoder
- Decoder

Which in this computer assignment we are not dealing with the $Output\ Embedding$ and the $Decoder$.

Input Embedding:

This part we are going to design a layer with 6 semi-layers containing a 2D Convolutional layer, Batch Normalization layer, HardTanh layer, 2D Convolutional layer, Batch Normalization layer and again a HardTanh layer.

For more information we the $Hardtanh(x)$ functions that the output of the function is the Figure below.

$$\text{HardTanh(x)} = \begin{cases} 1 & if\ x > 0 \\ -1 & if\ x < -1 \\ x & otherwise \end{cases}$$

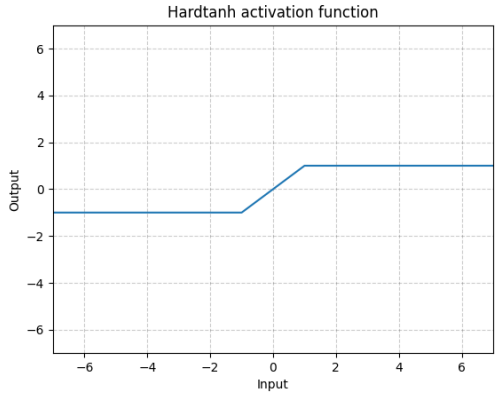

Figure 2 HardTanh

Figure 2 show the plot of the HardTanh(x) function as you can see it looks like the $\tanh(x)$ function.

Similarly, to the sequence transduction model, we use leaned embedding to convert the input tokens and output tokens to vectors of dimension d_{model}. We also use the usual learned linear transformation and Softmax function to convert the decoder output to predicted next-token probabilities. In our model, we share the same weight matrix between the two embedding layers and the pre-Softmax linear transformation. In the embedding layer, we multiply those weights by $\sqrt{d_{model}}$.

Position Encoding:

Since our model contains no recurrence and no convolution, in order for the model to make use of the order of the sequence, we must inject some information about the relative or absolute position of the tokens in the sequence. In this layer are going to use the Sin and Cos function for encoding the position. In this work we use sine and cosine functions with different frequencies.

$$PE_{pos,2i} = \sin\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right)$$

$$PE_{pos,2i+1} = \cos\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right)$$

Where in the following equations the parameter pos is the position and the i is the dimension, that is each dimension of the position encoding corresponds to a sinusoid. The wavelength geometric progression from 2π to 20000π. The reason that is the paper they used this function is because the hypothesized would allow the model to easily learn to attend by relative position, since for fixed offset k, PE_{pos+k} can be represented as a linear function of PE_{pos}. We done all of the following calculation in the Class layer in Python.

Encoder:

The next part of the model it may be the most important part of the network it contains a Dropout layer where the probability of the dropout is 0.1 and there is a linear layer, and a Layer Norm after that is the Position Encoding layer where we talked about in the previous part and we already know all about the equations of is and the reason for every part and after that we use 4 sequence Encoder Layers and we are going to talk about this layer in the following parts. The following parts are in this layer but we are going to talk more about them.

And we also need to mention that we added an output layer with a Linear and LogSoftmax layer to get a proper output that we want the linear layer is for converting the dimensions to each other and the LogSoftmax is a layer to get a reasonable score in the output and a non-linear part in the output so we can calculate the gradient easily, and if we used a different non-linear part in an output such as a regular Softmax function we will get different output from the model.

Encoder Layer:

In this part the model contains two part called the $Self\ Attention$ and the $Positionsize\ Feed\ Forward\ With\ Conv$ which is going to be covered more significantly in the future.

The encoder is composed of a stack of N = 4 identical layers. Each layer has two sub-layers. The first is a multi-head self-attention mechanism, and the second is a simple, position-wise fully connected feed-forward network. We employ a residual connection around each of the two sub-layers, followed by layer normalization. That is, the output of each sub-layer is LayerNorm (x + Sublayer(x)), where Sublayer(x) is the function implemented by the sub-layer itself. To facilitate these residual connections, all sub-layers in the model, as well as the embedding layers, produce outputs of dimension $d_{model} = 512$.

Self-Attention(MultiHeadAttention):

In this Layer we have a large number of layer which each have a reason for being. At the beginning we have 3 Linear layers which we call $Query\ Linear$, $Key\ Linear$ and the $Value\ Layer$ where all the dimensions of the this three layer are the same as each other and after that there is a layer called the $ScaledDotProductAttention$ layer which we are going to talk about in the future and after that there Layer Norm layer and another Linear layer and at the end there is dropout so we can try to generalize the network more efficient in this case.

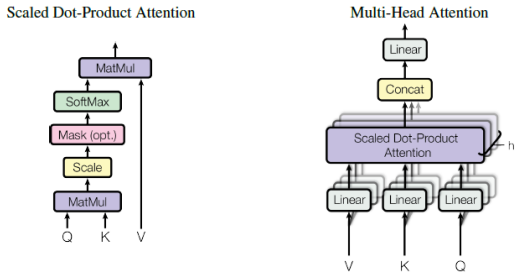

Figure 3

Figure 3 shows the structure of the model as a picture, we can see the Multi-Head Attention model the structure that we send the three parameters to the network and all this three are the same in our network so we don't have a problem with their dimensions after that we send it to the Scaled-Dot-Product and after concatenating the results we send it to a linear layer and next to the output.

Instead of performing a single attention function with d_{model}-dimensional keys, values and queries we found it beneficial to linearly project the queries, keys and values h times different learned linear projections to d_k, d_k and d_v dimension, respectively. On each of these projected versions of queries, keys and values we then perform the attention function in parallel, yielding d_v-dimensional output values. These are concatenated and once again projected, resulting in the final values, as depicted in Figure 3.

$$MultiHead(Q, K, V) = Concat(head_1, \dots, head_h)W^Q$$

$$where\ head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

There formulas are taken from the uploaded paper in the computer assignment.

Attention(ScaledDotProductAttention):

In this layer there is only two independent layers such as a dropout layer and a Softmax layer. In practice, we compute the attention function on a set of queries simultaneously, packed together into a matrix Q. The keys and values are also packed together into matrices K and V. We compute the matrix of outputs as:

$$Attention(Q, K, V) = Softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

The two most commonly used attention function are addictive attention and dot product (multiplicative) attention. Dot-product attention is identical to our algorithm, except for the scaling factor of $\frac{1}{\sqrt{d_k}}$, Additive attention computes the compatibility function using a feed-forward network with a single hidden layer. While the two are similar in theoretical complexity, dot-product attention is much faster and more space-efficient in practice, since it can be implemented using highly optimized matrix multiplication code. While for small values of d_k the two mechanisms perform similarly, additive attention outperforms dot product attention without scaling for larger values of d_k. We suspect that for large values of d_k, the dot products grow large in magnitude, pushing the Softmax function into regions where it has extremely small gradients. To counteract this effect, we scale the dot products by$\frac{1}{\sqrt{d_k}}$.

POS_FFN(PositionwiseFeedForwardWithConv):

In this layer we are going to use 4 simple layer such as two Convolutional layer and a dropout and Layer Norm layer for this part which in the convolutional part we also use the padding and the stride. In addition to attention sub-layers, each of the layers in our encoder and decoder

contains a fully connected feed-forward network, which is applied to each position separately and identically. This consists of two linear transformations with a $ReLU$ activation in between.

$$FFN(x) = \max(0, xW_1 + b_1)\, W_2 + b_2$$

While the linear transformations are the same across different positions, they use different parameters from layer to layer. Another way of describing this is as two convolutions with kernel size 1. The dimensionality of input and output is $d_{model} = 512$, and the inner-layer has dimensionality $d_{ff} = 2048$.

The Loss Function:

In this part we are going to talk about the type of loss function that we used in this network. The loss function that was used is called $CTC\ Loss$ or in other words Connectionist temporal classification.

Connectionist temporal classification (CTC) is a type of neural network output and associated scoring function, for training recurrent neural networks (RNNs) such as LSTM networks to tackle sequence problems where the timing is variable. It can be used for tasks like on-line handwriting recognition or recognizing phonemes in speech audio. CTC refers to the outputs and scoring, and is independent of the underlying neural network structure. It was introduced in 2006. The input is a sequence of observations, and the outputs are a sequence of labels, which can include blank outputs. The difficulty of training comes from there being many more observations than there are labels. For example, in speech audio there can be multiple time slices which correspond to a single phoneme. Since we don't know the alignment of the observed sequence with the target labels we predict a probability distribution at each time step. A CTC network has a continuous output (e.g. Softmax), which is fitted through training to model the probability of a label. CTC does not attempt to learn boundaries and timings: Label sequences are considered equivalent if they differ only in alignment, ignoring blanks. Equivalent label sequences can occur in many ways – which makes scoring a non-trivial task, but there is an efficient forward–backward algorithm for that. CTC scores can then be used with the back-propagation algorithm to update the neural network weights.

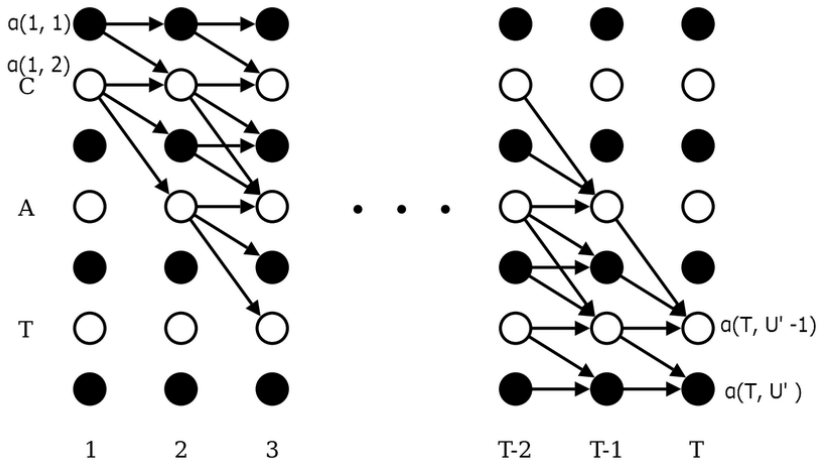

Figure 4

Figure 4 shows the result about CTC table and shows the result and we can see how the calculations are going this loss class in defined like below for more information.

$$Class\ torch.nn.CTCLoss(blank = 0, reduction = "mean", zero\_infinity = False)$$

Calculates loss between a continuous (unsegmented) time series and a target sequence. $CTCLoss$ sums over the probability of possible alignments of input to target, producing a loss value which is differentiable with respect to each input node. The alignment of input to target is assumed to be "many-to-one", which limits the length of the target sequence such that it must be $\leq$ the input length.

$$p(I|x) = \sum_{\pi \in \mathrm{B}^{-1}(I)} p(\pi|x)$$

$$\alpha_t(s) \stackrel{\text{def}}{=} \sum_{\pi \in N^T, \beta(\pi_{i;t})} \prod_{t'=1}^{t} y_{\pi_t'}^{t'}$$

$$\alpha_1(1) = y_b^1, \alpha_1(2) = y_{I_1}^1, \alpha_1(s) = 0, \forall s > 2$$

$$a_t(s) = \begin{cases} \bar{\alpha}(s) y_{I'_s}^t & if\ I'_s = b\ or\ I'_{s-2} = I_s{}' \\ \left(\bar{\alpha}_t(s) + \alpha_{t-1}(s-2)\right) y_{I_s}'{}^t & otherwise \end{cases}$$

$$where\ \bar{\alpha}(s) \stackrel{\text{def}}{=} \alpha_{t-1}(s) + \alpha_{t-1}(s-1)$$

$$p(I|x) = \alpha_T(|I'|) + \alpha_T(|I'|-1)$$

$$\beta_t(s) \stackrel{\text{def}}{=} \sum_{\pi \in N^T, \beta(\pi_{i;t})} \prod_{t'=1}^{t} y_{\pi_t'}^{t'}$$

$$\beta_T(|I'|) = y_b^T, \beta_T(|I'|-1) = y_I^T, \beta_T(s) = 0, \forall s < |I'|-1$$

$$\beta_t(s) = \begin{cases} \bar{\beta}_t(s) y_{I'_s}^t & if\ I'_s = b\ or\ I'_{s-2} = I_s{}' \\ \left(\bar{\beta}_t(s) + \beta_{t-1}(s-2)\right) y_{I_s}'{}^t & otherwise \end{cases}$$

$$\bar{\beta}(s) \stackrel{\text{def}}{=} \beta_{t+1}(s) + \beta_{t+1}(s+1)$$

Optimizer:

In this assignment we are going to use the Adam optimizer the following equations for it and each parameter ω^j

$$CLASS\ torch.optim.Adam(params, lr = 0.001, betas = (0.9, 0.999), eps = 1e-08, weight\_decay = 0, amsgrad = False)$$

$$v_t = \beta_1 * v_{t-1} - (1 - \beta_1) * g_t$$
$$s_t = \beta_2 * s_{t-1} - (1 - \beta_2) * g_t^2$$

$$\Delta\omega_t = -\eta \frac{v_t}{\sqrt{s_t + \epsilon}}$$

$\eta: initial\ Learning\ rate$
$G_t: Gradient\ at\ time\ t\ along\ \omega^j$
$v_t: Exponential\ Average\ of\ gradients\ along\ \omega_j$
$s_t: Exponential\ Average\ of\ squares\ of\ gradient\ along\ \omega_j$
$\beta_1, \beta_2: Hyperparameters$

And also we used a changing learning rate for the classification.

$$Learning - rate = d_{molel}^{-0.5} \times \min(StepNum^{-0.5}, StepNum.WarmUp^{-1.5})$$

Where the parameter $WarmUp$ in out equation is 4000.

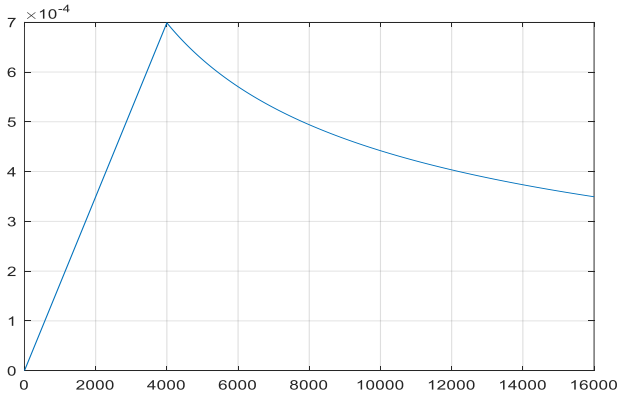

Figure 5

Figure 5 shows the result of the Learning rate depending on the number of steps this shows that at the learning rate function we can see the first steps have a small size and the next steps get much bigger until we reach $step\_num = WarnUp$ and after that the learning rate will start to decrease by time and that suits us as well because from a step beyond we need to decrease the learning rate in case to reach a good local minimum.

Part 2

1) In This part we are going to show the result of the network that we have implemented in the previous part where we used the prepared data from the dataset and before that we toke the spectrogram from the data.

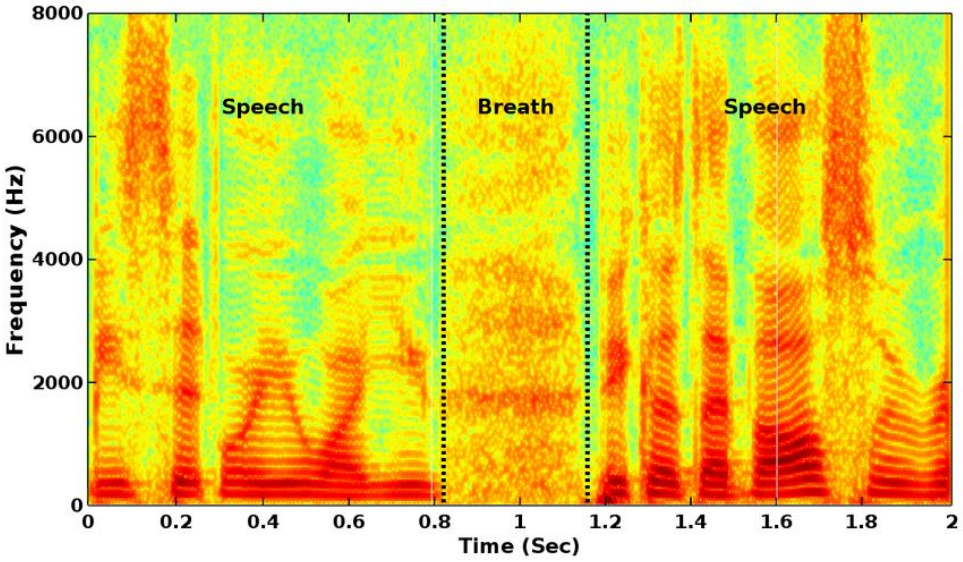

Figure 6 Spectrogram sample

A spectrogram is a visual representation of the spectrum of frequencies of a signal as it varies with time. When applied to an audio signal, spectrograms are sometimes called sonography, voiceprints, or voice grams. When the data is represented in a 3D plot they may be called waterfalls.

This part of the network was being implemented before by the code in the uploaded files. And After that we feed the network with the mentioned data from our dataset and got a loss and by using that loss we trained our network and updated the weights of the network.

We used three methods to compare the result of the training in the training and validation dataset where we are going to see the result in the next part.

Method Loss:

We have already talked about the loss function and the formula of it and there is no need to explain more, we can see the result in Figure 7 and 8 where the loss of the train data and the validation data start to decrease by time for the train loss we took a sample for every 200 steps but in the validation dataset we took the data of 3 batches but we didn't use them for the updating the network so we can say that they were validation dataset so as we see we done it about 19 epochs the loss decreases as we expect but because of the large dataset that we are using and the number of parameters the network didn't get over fitted.

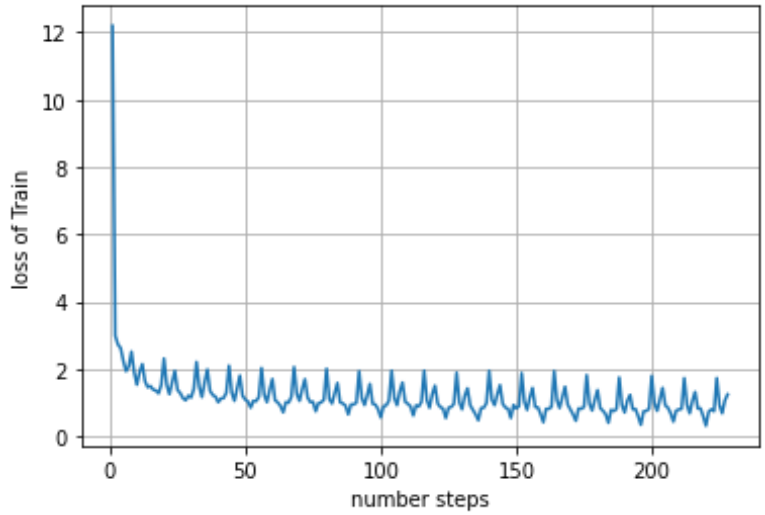

Figure 7 Loss of train

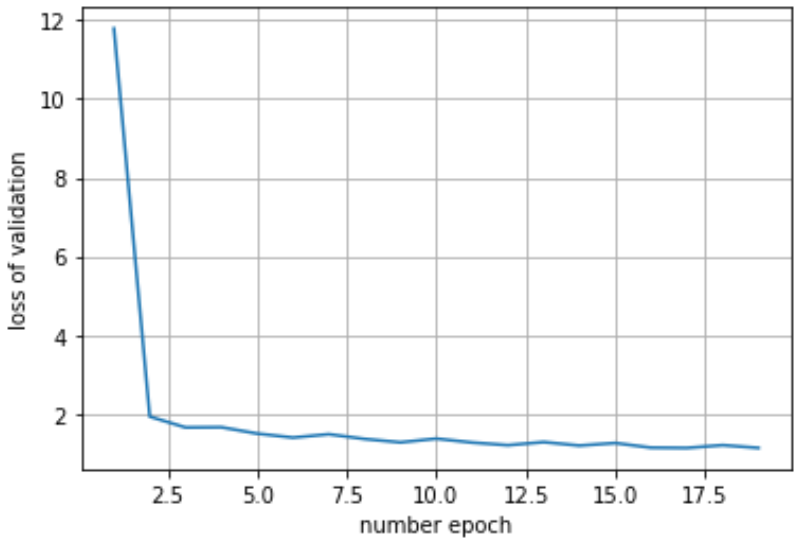

Figure 8 Loss of Validation

Method CER:

This method is called the character error rate where in that we get the two sentence of the input and the output of the network and we calculate the Levenshtein distance of the two strings and is the minimum number of changes needed to make a the two strings the same.

We are going to use a particular distance for the equations, it's called the $Levenshtein\ distance$ and it calculates the distance of two strings a, b where we represent the length of the strings as $|a|, |b|$ and the distance is given.

$$lev_{a,b} = \begin{cases} \max(i,j) & if\ \min(i,j) = 0 \\ min \begin{cases} lev_{a,b}(i-1,j) + 1 \\ lev_{a,b}(i,j-1) + 1 \\ lev_{a,b}(i-1,j-1) + 1\_(a_i \neq b_j) \end{cases} & otherwise \end{cases}$$

In information theory, linguistics and computer science, the Levenshtein distance is a string metric for measuring the difference between two sequences. Informally, the Levenshtein distance between two words is the minimum number of single-character edits (insertions, deletions or substitutions) required to change one word into the other. It is named after the Soviet mathematician Vladimir Levenshtein, who considered this distance in 1965.

So by using this method an easier formula to calculate this parameter.

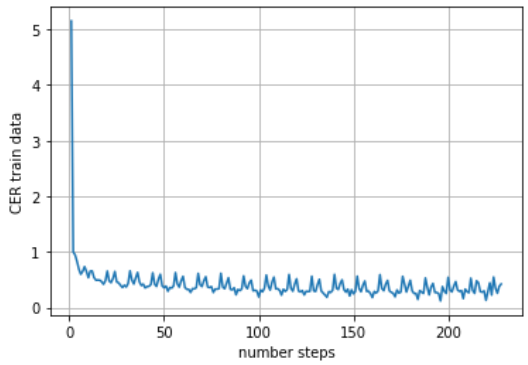

Figure 9 CER Train

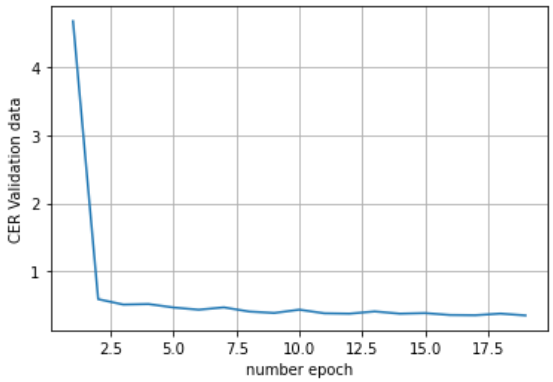

Figure 10 CER Validation

Method WER:

This part is just like the previous part about CER but the difference is that it will calculate the distance of word than characters in the past part.

According to https://en.wikipedia.org/wiki/Word_error_rate the formula of WER is:

$$WER = \frac{S + D + I}{N} = \frac{S + D + I}{S + D + C}$$

- S is the number of substitution,
- D is the number of deletions.
- I is the number of insertions,
- C is the number of correct words,
- N is the number of words in the reference (N=S+D+C)

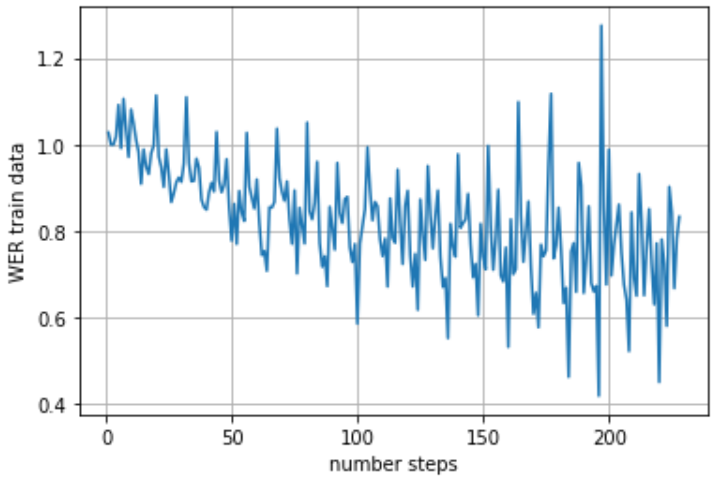

Figure 11 WER Train

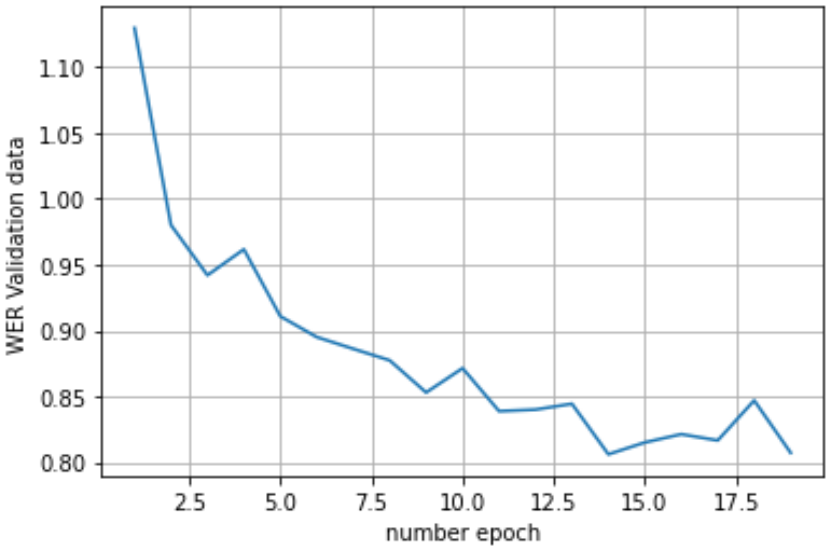

Figure 12 WER Validation

2) In this part we are going to show 12 sentence in the model where we show the output of the network, it needs to be mentioned that we have cleaned the output which means if there are characters which we are not pleasant with are removed.

```
1 PAD_CHAR = "ʗ"
2 SOS_CHAR = "§"
3 EOS_CHAR = "¤"
```

Figure 13

Figure 13 shows the characters which all have meaning for padding and between he words but don't have any meaning to us are removed from us to see and we also got rid of characters where there is character that is being repeated, the result is going to be showed below and the text has been uploaded as well with the Code and this document.

Result 1:

mosor teras o te o e seos an ofis besoio en

ofesjrons ermesacncerde beison an matese

ine theanes ora nor o woso fisin teseme fet moe

theres or be in tes thede mere ferstea

no idse boes weren er mene servens

he ised core aonsen e mery mao wos

r joiserer theoer manasdoartiel

i rotrelo wretes li mensteashnesenstnes

anderse fhero lisen de the bo ner wer

ba her om monsarses ne os bego

we mo gef arsos an no ore mone olo

te lons woer no bereo

Result 2:

mucsh mar bernisto bhem othem ses an o fis disilosinmend

odefestresions ersa cnjerdefesion an mateset

anethe insy ore a norma was fision tsenm e felot molei

thers morbe in tis thenanm mere furst tel

no idis bedlis werin thermeny seronst

bheontiscud cory ucinsin ha maryt mabo wos

your jome sr theaer managt ertictlad

ne rotre pli writhe sli mene stagne senstof es
anterce ferce lis in tetha bo never wer od
but r m muntarses letus egot
we mus gi erses and not or mouni olo
the lins wer nowverytod

Result 3:

much mor ernis tol them o themses and ofis tisilosionmemd
afestretions ern iset cnjerdefision and ma temse
ani thig inusu are a nor ma waso fisioin tosem i feo t moleit
theris orebe in tis then an mereiyun e ferste itel
no itisa beles rerin therer many serfens
beyon dised cory hutinsin a maryt maba aos
yurn gn mesere the alter man estert i clad
ne rotre lit rithe li nen stegsino senstef es
andteshe fer se lis in thet te bil neverwar od
but her o munsar sels let us begd
we mus givf ar ses and not are moneg ilo
the lins were now veresod

Result 4:

much mar ernis to them of themsels and ofhis dislosionend
orfep stretions ernesaet onjerd ofion ad ma themsle
any the in usu or a nremo wasofition tosem he felot moltici
theris mare be in this thanang mergunefrs te ite
noidisa bates rerin ther menyservens
ean tispud cory hacinson had meryt maba wols
your jonmeser te other manax tur te clad
e routre plid writhe sli menef stasion osenstofnes
andterche fer to tis ind the thy bol never werod
but her o mnsar sels lhet s begod
we mus giv arses and not or monen olo
the lins were noverytad

Result 5:

much mor bernis jal them ot themsels and of his disilosionment
otfepstrections ernisat condturdofition and ma themsed
anythin anusu wor anormo was fition tosen the felo tomotaci
theris ore be hin tis then en meruni firste ite
no i isap balis rerin ther a many sertens
beon tisput cor hucinsin had maret mabl wos
you r jokn me sir the ather manexto werticlad
ne rotre pliat re the slihtmenestagin o senstofnes
andter che fur cotis in thet they bo never werod
but her om munsarsels let us begod
we must give arses and not or muny olon
te lins were novery ta

Result 6:

muchor erns to thom of themses ando his tisosion end
ototestretions berne sat conjerd ofition and ma themse
anye thin inusuo or nor ma waso fison tosente felo to motici
ther is mare be hine is thenem meragunifrste te
no itis a bats werin therer meneservens
be on tisud cory hucinson hat mryet mabal wos
yo re jong me sir theother manes tu erticlat
ne rotreplit write sli mene statieno senstofnes
andtercef fer co this an thet the bo never werod
but her a munsarsols tet uspegot
we mus divf erses and not or mony olog
the lins were now vere tot

Result 7:

much mor ernes to tem of thmselvs an of his tisiloutionment
aotofpstrecions ernes hat conterdof fition and ma them sed
anythin ins ora normo waso fition tosente felo to motei
theris orebe hin tis thenang mer uni frs te it

no id isa badis werin ther meny servens
be on tispud cory huthinson had maret maba wos
you re jo me sirn theother manaxe d rtoclad
nem rotre plit wethe stlih meneo statin o senstofnes
and ter ce ver cutis ind that they bi thaverwerod
but her a munstarsalv let a spegod
we mus give arsels and not or mun e olong
the lins were knowvere tad

Result 8:

much mor gerns to them of thamselvs and of his disalusionmend
otofepstrecsions ernes hat conjurd ofision and mad themsed
any thin an u su ore ap normo wasof fisin tosem he felo t motaci
there is more be hinthis thetn a mera yunafirs te ite
no it isa batits rerin there r menty serfens
beyon tespud cory hachinson had maryt mabl wons
yo wre jukg me sir the other manex u articlat
nemrotry pliet re he slit meni stacion n o senst fnes
and ter chefe fur tu lis ind that they bo nhaver werot
but here amonstarselvs letus begout
we mos give orses and not or mony o long
the lins were now verytad

Result 9:

much mor ernistoa them o themsevs and of his disolusionmend
odtapstrections ernisheat congerd of vision and mad themsed
anythig ausu ara nmoarma wasofision tosenefelo to moleci
ther is mor be hin this then in meri un ifirs te ite
no it is a batis werin ther menyserfens
be on tispud cory hucnson had mart mava wans
you n jonme sir the other mantstu artilat
nem rotre plid rite sligh menifo stasin of senst ofnis
and ter chef fur t tis ind that the bl naver werod

but her a mongsarsols let uspegout
we mus give arses and nut or muni elon
the lins were novere tat

Result 10:

much mor ernis od them of themsels and of his thisalusionend
otitepstrections ernas hat contured of fision and mayd them sed
any thin inusu or ap noremo wasufisin tosem the felo to moteci
there is more behin this then a merea une firste ide
no it isa batits werin ther r meny serfenc
beon ti spud gary hatinson had mary mabl wos
you re joknme sir the other manistoarticolatd
nim rotry plit rithe slightmenie stagin of senstofnes
and ter chev fir cu lis in that they wil never werowtn
but her a mynsar selves let u sbegot
we muts giv erselvesand not or monyg olong
the lins were now very talt

Result 11:

much mor ernis to them of themselvs and of his disalusionmend
otifapstrections ernis hap cansurd ofision and made themsend
andy thing anusu orap normo wasofisio tosen i felo to motici
there is more be hin this thet ne mere yunifirsty itel
now it isabalis merin there ar r meny servens
be on tispud gory hacinson had marit mabal wos
you rnt jkinme sir the oter manids tyou o ticlad
nim rotry pliet writ a slight nenieistasion of senstofines
and ter che vir cu lis ind that they bi never werod
but her a mongstarselves let u spegoed
we must give ourselves and not or monyg olong
tho lins were now very tod

Result 12:

much mor ernis tol them of themselvs and of his dislusionmend

outofepstrections ernishat conjerdo vision and mad temsel
andyti inusu ori nr mo wasofisio tosentefelo to molei
there is more be hin tis then a mereayunifirsto ite
no it isapalis werin ther meny serviens
beyon tispud cory hutinson had maryt mabal was
yo are jokg me ser the other manig jo r ticolat
i rotry plid wit e slightmeniestasin o senstofmes
and trhefe ver cu i s in tat thay bil never wero
but here a monsarselves let uspegou
we must giv orselves and not or mony a lon
tho liens were nowverytald

Result 13:

much mor ernis tod them of themselves and of his thisilusionmend
outotepstrections ernis hav conjerd o fition and mad them send
anything anusu orap normo was o fition to sene felo to moteci
thereris more be hin this thanim mere unefirsty ite
no it is a batits werin there ror menyservens
be on tispud cory hachonsin had marit mabl wons
you arn j okingme sir the other manas to ar ticlate
nem rotre plin rithe slike nenitestation o senstofnes
and ter cev ver tu lis ind that the blil naver weroud
but here a monstaurselves lat ous pegoud
we must give ourselves and not ouer monig lon
the lins were now very talde

Result 14:

much mor ernis to them of themsels and of his dhislusiond
oudifapstrections ernis hav conurdifisin and mad temsed
anything ius ora normo wasof fison tosen e feo to motei
thereis more be hindtis tha a merigunifirste ite
no it isa peatis werin there r meny servins
beon dispud cory huconson hade maryt mabal ols

you r joigme sir deother manitg orticliat
em rotry lid riti slighe menifestasion o senstofnes
and ter chef ver tu tis and that they bil never werod
but her amonsor selves lat osbegod
we mus give ourselves and not or mony a long
te lins were now verytad

Result 15:

much mor ernis to them of thimselvs and af his thisalusionend
ot ofpstrectsions ernis had conjurd ofision and made temset
andy thing iusu or i norma was ofision tosendo felow to moleci
thereis more behinthis thana marea unifirs ty ite
now it isa padis werin there r menyservns
be on thespud cory hacnsin hade maryd mabl wols
you are jok me sir the other manig dyou ar ticulat
himrotry plind wit hi slighe manifistasion of senstofnes
and ter chef vir tu dise ind that they bil never werond
but here a mungsarselves let o s pegod
we must dive arselves and not our mony olon
to lines were nowveryt tad

Result 16:

much mor ernis to them of themselves and of his this lusionmend
out ofepstrections eirnis hat conerd ofision and mad thempsed
any thin iusu or ap norma wa sofisien toseno felo to moicy
there is more behind this then an mere unifirs to ide
no it isapatic werer in there re menyserfence
be on dispud cory huchonson had maryd mabal wols
yu are jokig me sir the other manage yo articulat
im rot ry plint with i slighknenifestasion o senst ofneso
and ter ef vir tu lis in that they bl never weront
but here a monsteurselves let uspegoute
we must give ourselves and not our mony a longv

the lin s were now verytate

Result 17:

much mor ernis tol them of themselves and of his dislusionmend
oud ifapstrections ernes hav conturd ofision and mad themse
andy thing iusual ora nor mo waso fision to sen o felo to molecys
there is more behin this than an mere yunfirsty ide
no it is bedit rererin there r manyservens
beyon thispud cory huchonson had maryt mavl wols
you re jokig me sir the other maneseto ar ti kulat
erotry pliede rit slightenenifestasion o senst ofnes
and ter cef vur cu lis in that they bi never werot
but her a monseurselves let u spegont
e must dive ourselves and not our mony alon
the lins were now very tate

Result 18:

much mor ernis told them of themselves and of his dhisalusionment
out ifapstrections ernes had condurd ofision and made themset
ande thin iusu or ap normo waso fision to sene felo to moliciy
there is more behindthis then in mere unefirsty ide
no it is a pale werin there e re meny servens
teyo tispud cury huch enson had mared mava was
yo u r joking me sir the other manage de a ticulate
himrotry plide rit e slighemenifstastion of senstofmes
and ter thef ver tu tis in that they bil never werond
but here a munseurselves letuspegout
we must giv ourselves and not our mony olon
the liens were now very tad

Result 19:

much more ernstod them of themselves and of his dhis alusionmendt
out if apstrections ernis had condurd ofision and made themcet
andething iusu or a normo was o fision to en o felo to moneci

there is more behinethis tha a mere i unafirsty itelo

no it isa palitc whererein there are meny seretenc

teyon tispud cury huchansin had mare mabl wals

you r jokig me sir the other manage o or ticulate

imrotre plide rit i slightnanstasiong of senst of fnes

and ter cev vur tu lie ind that they bil never weround

but here a monseurselves let uspegout

we must give ourselves and not our muny o longngete

the lines were now verytate

As we can see the result improves in every epoch of the training so it shows that out network is working correctly and that is the main goal of this computer assignment.

We can believe that we have done pretty well on it because the main transformer in the paper had a decoder in it and the amount of data for that we even larger and it has been trained on 8 GPU for more than two days the following result is from a smaller sized dataset and is was trained on google Colab server for about 8 hours with lots of problems with the GPU memory, and the crashing GPU when we used big data.

We also didn't use big data in this computer assignment because of the size the GPU started to crash so if the input data had a dimension bigger than 1000 we just skipped that data and went to the next one.

Best path decoding:

Best path decoding is the simplest method to decode the output matrix:

- Concatenate most probable character per time-step which yields the best path
- Then, undo the encoding by first removing duplicate characters and then removing all blanks. This gives the recognized text.

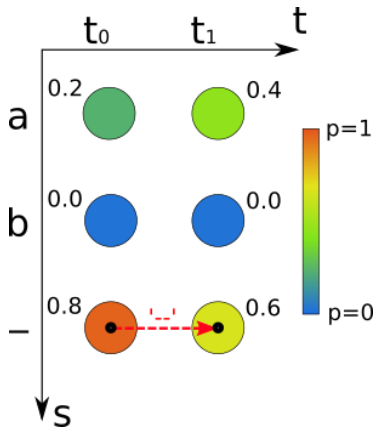

Figure 14

Let’s look at an example: the matrix is shown in Fig. 14. The highest-scoring character is blank for both time-steps t0 and t1. So, the best path is ‘--’. We then undo the encoding and get the text “”. Further, we can calculate the probability of the path by multiplying the character-probabilities, which is 0.8·0.6=0.48 in this example.

Best path decoding is fast; we only have to find the character with the highest score for each time-step. If we have C characters and T time-steps, the algorithm has a running time of O(T·C).

Why best path decoding can fail:

Best path decoding is both fast and simple, which are of course nice properties. But it may fail in certain situations like the one shown in Fig 14. In Fig. 15 all paths corresponding to the text “a” are shown: ‘aa’, ‘a-’ and ‘-a’. The probability of the text “a” is the sum over all probabilities of these mentioned paths: $0.2 \cdot 0.4 + 0.2 \cdot 0.6 + 0.8 \cdot 0.4 = 0.52$. So, “a” is more probable than “” ($0.52 > 0.48$). We need a better algorithm than best path decoding which can handle such situations.

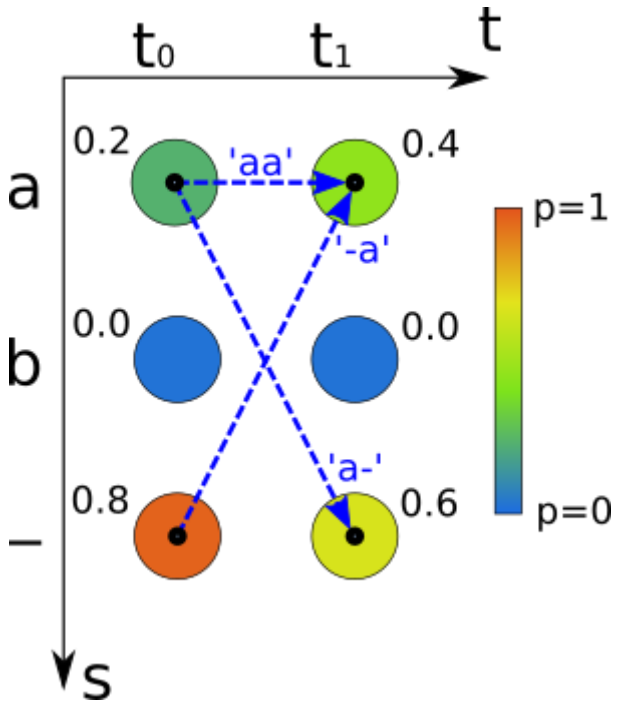

Figure 15

Process

In this computer assignment we leaned many things from handling bugs to learning about transformer networks, as it was said we first needed to load the large dataset using a different environment after that we needed in implement the embedding layers of the network and after that a number of other layers where needed for the task but thank God there was no need to implement the decoder for the network because that part was pretty hard as well. The network that we were using in a kind of end-to-end network, where there is no recurrent network in it and is an up to date method for networks, we used the CTC loss functions as well with e changing learning rate, one of the hardest part of the project was to find out was the meaning of the output of the model there has a high dimensional tensor in the output and the way that it needed to work with the labels and all that wasn't easy at all. At the end of the report we showed the output results of the network and it was nice knowing that how this transformer work and how to they get trained in order to convert the speech to text.

Reference

https://github.com/Holmeyoung/crnn-pytorch

https://discuss.pytorch.org/t/ctcloss-performance-of-pytorch-1-0-0/27524/2

https://github.com/SeanNaren/deepspeech.pytorch/blob/81cb5755c08d6e90abc6975dcfc4bc5a89bdfb6a/decoder.py#L43

https://github.com/gentaiscool/end2end-asr-pytorch/blob/a22efdda28f1689206eefb3fedbb56fbcc98a753/utils/metrics.py#L58

https://github.com/SeanNaren/deepspeech.pytorch/blob/81cb5755c08d6e90abc6975dcfc4bc5a89bdfb6a/decoder.py#L43

https://colab.research.google.com/drive/1IPpwx4rX32rqHKpLz7dc8sOKspUa-YKO#scrollTo=RVJs4Bk8FjjO

https://github.com/gentaiscool/end2end-asr-pytorch/blob/master/models/asr/transformer.py

https://github.com/gentaiscool/end2end-asr-pytorch/blob/master/models/common_layers.py

https://github.com/gentaiscool/end2end-asr-pytorch

https://holianh.github.io/portfolio/Cach-tinh-WER/

https://en.wikipedia.org/wiki/Levenshtein_distance

https://www.rev.com/blog/resources/what-is-wer-what-does-word-error-rate-mean#:~:text=Basically%2C%20WER%20is%20the%20number,The%20result%20is%20the%20WER.

Attention Is all You Need, Ashish Vaswani, Noam Shazeer, Niki Pamar, jakob Uszkoreit, Llion Jones, Aidan N,Gomez, Lukasz Kaiser

Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks, Alex Graves, Santiago Fernandez, Faustino Gomez, Jurgen Schmidhuber