Final Project: Introduction to Speech and Audio Proessing

Tamuz Gitler 312243439

Asaf Ventura 313177404

Final model:

Our final model is a twist on the DeepSpeech Model:

The model architecture is composed by one convolutional layer for feature extraction (given mfcc), continue by 4 residual convolutional layers (uses skip connection), 3 bidirectional GRU (type of RNN) and a fully connected layer to shape the output as a ctc table input.

This architecture is designed to capture both local and temporal dependencies within audio data (this dependencies are crucial for understanding linguistic content, such as phonemes and words, which are structured in sequences.

Configuration:

Epoch	Batch Size	Optimizer	Learning Rate	Dropout (after conv)	Masking prob	Time Masking	Freq Masking
200	16	Adam	0.0005	0.1	0.4	20	10

Its performance is:

	Train	Validation	Test
WER	0.09921	0.1547	0.152
CER	0.0614	0.07	0.0569

Link to our model weights:

https://drive.google.com/file/d/1F6HSXav99xrO7wQJXESHXvAAlm5FIR-B/view?usp=drive_link

Below we describe our research work in details, for each model configuration we documented the WER and CER graph loss on the train and validation sets.

After training we documented the test WER and CER.

Our research was focused on three different phases.

Phase 1:

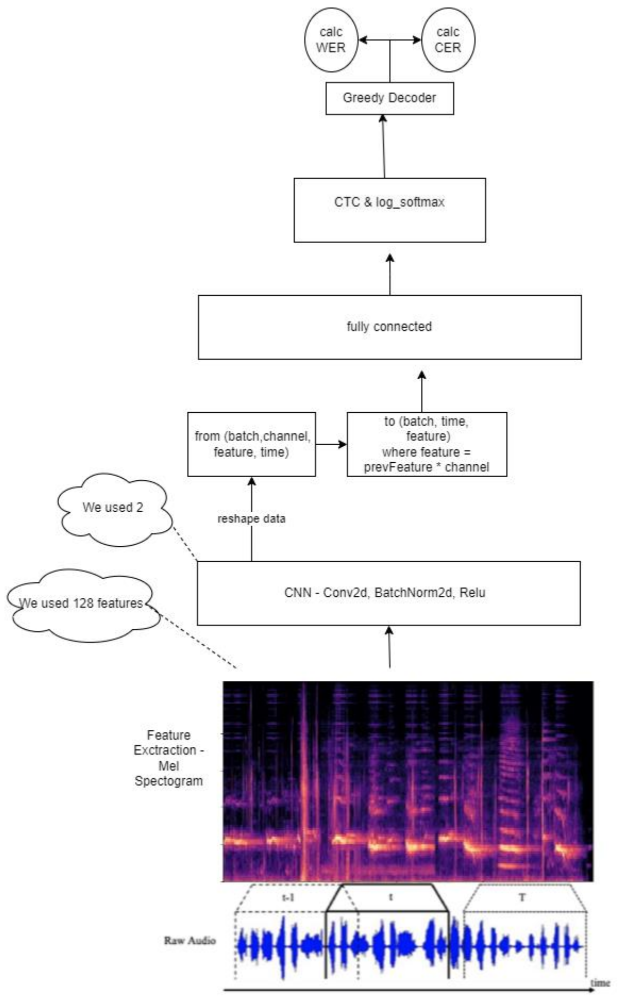
We started with exploring the most naïve baseline model:

CNN model: using 2 layers of conv2d, BatchNorm2d, ReLU following by a fully connected layer.

Conclusion from this phase:

Run number 2 achieved the best WER – 95%.

PipeLine

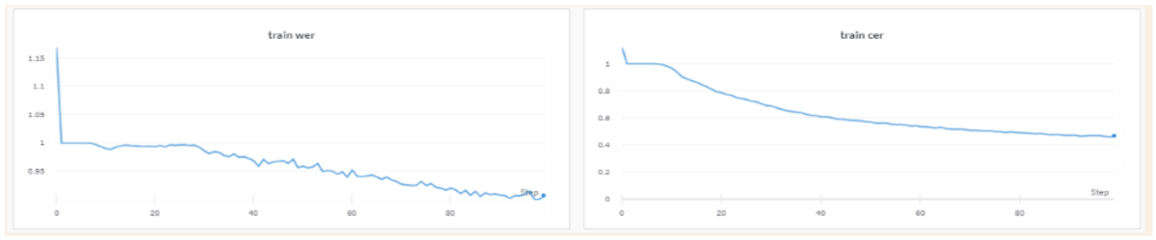

calc WER
calc CER
Greedy Decoder
CTC & log_softmax
fully connected
from (batch,channel, feature, time)
to (batch, time, feature) where feature = prevFeature * channel
We used 2
reshape data
We used 128 features
CNN - Conv2d, BatchNorm2d, Relu
Feature Exctraction - Mel Spectogram
t-1
t
T
Raw Audio
time

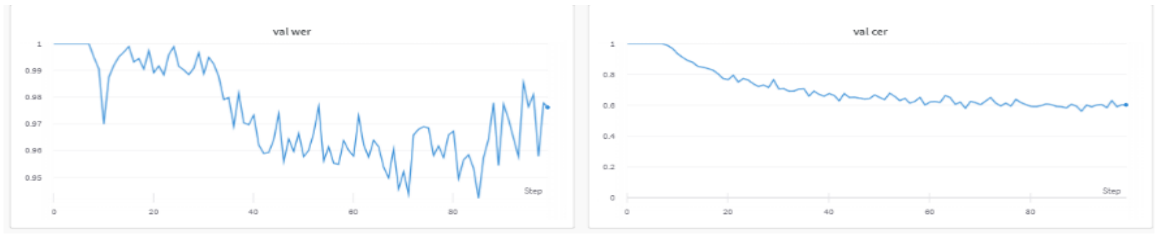
Run 1

Configuration:

Epoch	Batch Size	Optimizer	Learning Rate
100	20	Adam	0.0005

Train and Validation Loss:





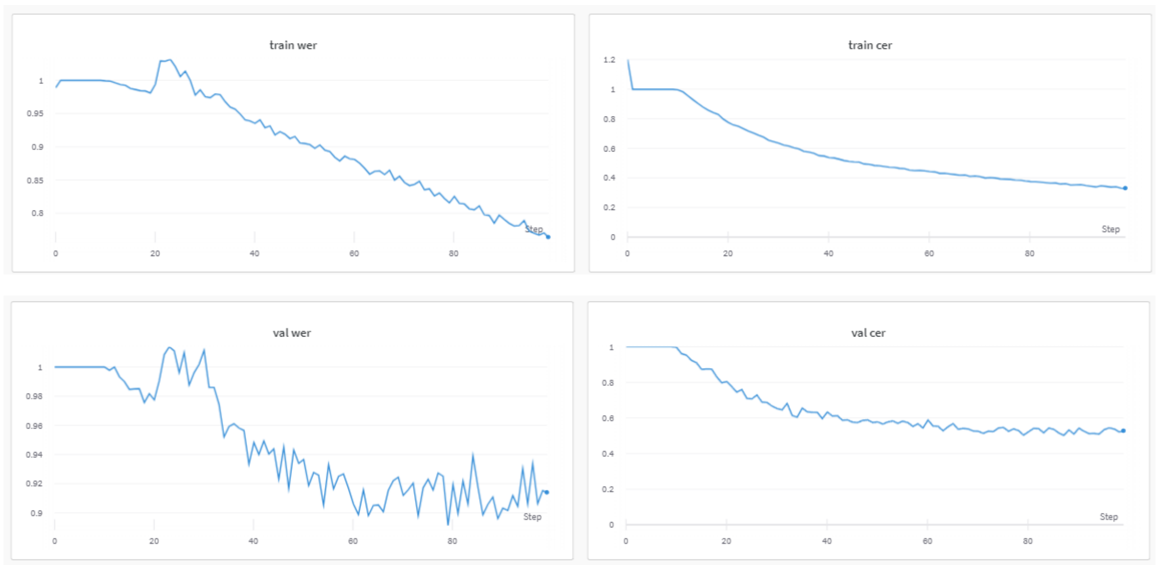
Test WER: 0.962

Test CER: 0.571

Run 2

Then we added another convolution layer to the model and used the same configuration.

Train and Validation Loss:



Test WER: 0.956
Test CER: 0.520

We managed to improve the loss a little bit, but 95% WER is was really bad.

Phase 2:

Next, following our findings in phase 1 we decided to implement more sophisticated network –

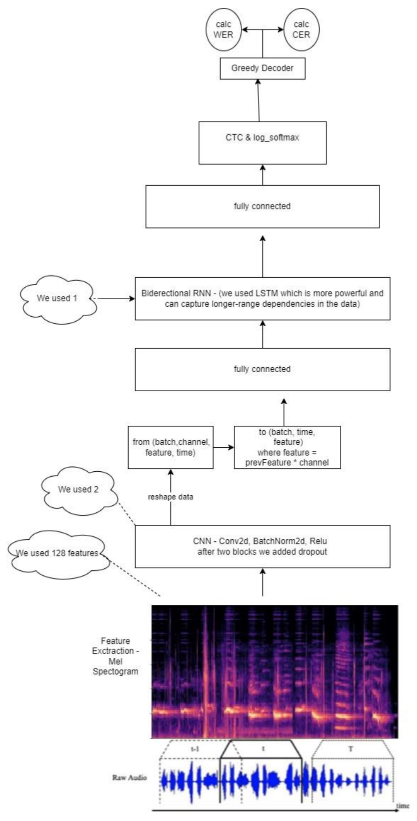
We used the previous model but decided to add <u>dropout</u> before the fully connected layer, followed by one layer of LSTM network due to its ability to effectively model sequential data and capture long range dependencies.

In this part we used different configurations:

Conclusion from this phase:

Run number 3 achieved the best WER – 31%.

PipeLine

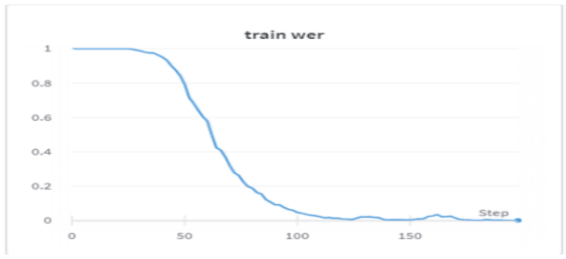

calc WER
calc CER
Greedy Decoder
CTC & log_softmax
fully connected
We used 1
Biderectional RNN - (we used LSTM which is more powerful and can capture longer-range dependencies in the data)
fully connected
from (batch,channel, feature, time)
to (batch, time, feature) where feature = prevFeature * channel
We used 2
reshape data
We used 128 features
CNN - Conv2d, BatchNorm2d, Relu after two blocks we added dropout
Feature Exctraction - Mel Spectogram
t-1
t
T
Raw Audio
time

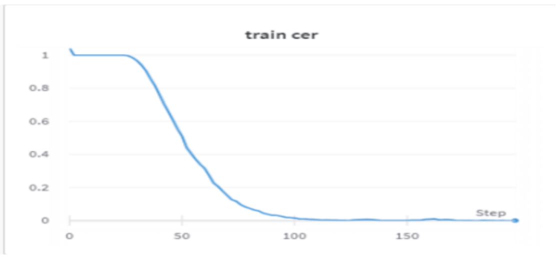
<u>Run 1</u>

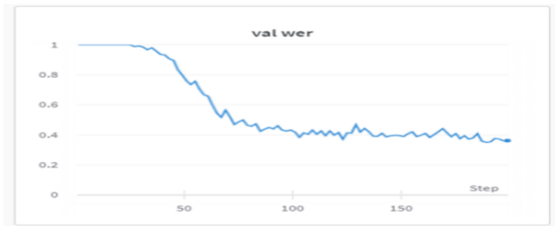
Configuration:

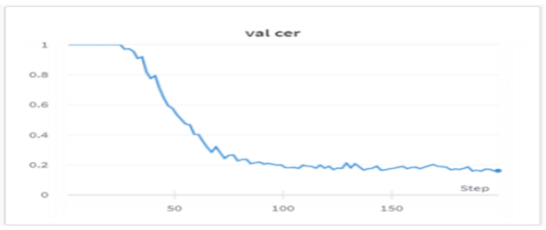
Epoch	Batch Size	Optimizer	Learning Rate	Dropout (after conv)
100	32	Adam	0.001	0.2

Train and Validation Loss:









test wer: 0.344

test cer: 0.152

This was an exciting results, we improved the previous test wer by 60% by adding dropout and LSTM!

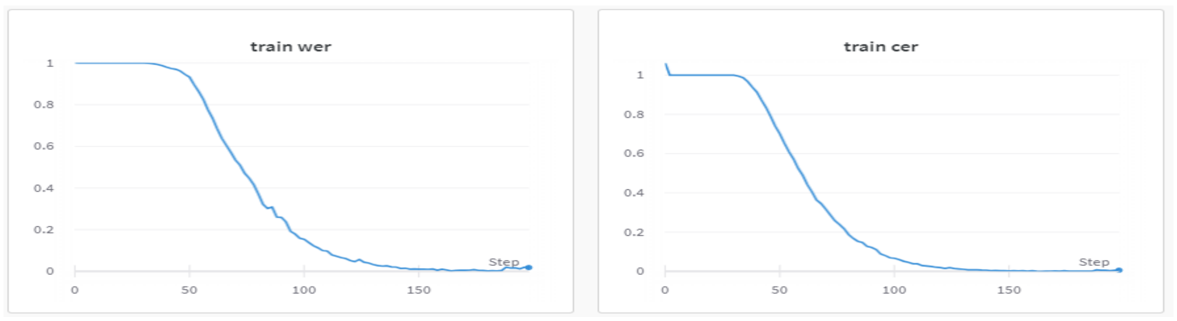
Run 2

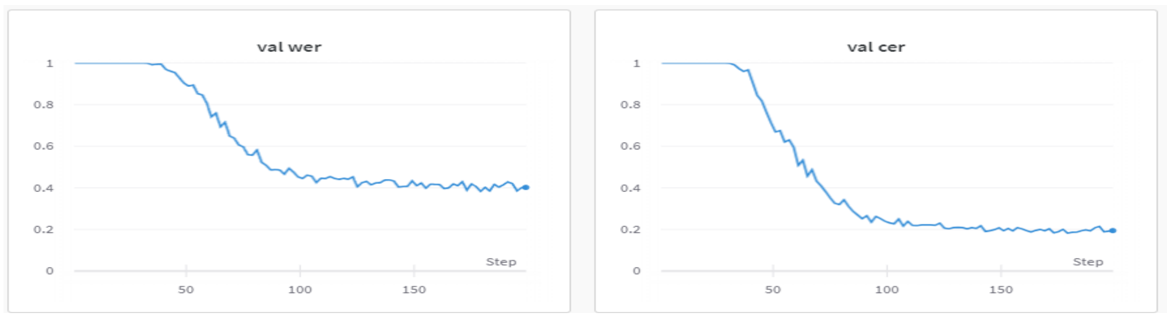
We wanted to explore the hyperparameters effect on the model, so we decided to changed the learning rate and the dropout values.

Configuration:

Epoch	Batch Size	Optimizer	Learning Rate	Dropout (after conv)
100	32	Adam	0.0005	0.1

Train and Validation Loss:





test wer: 0.399

test cer: 0.179

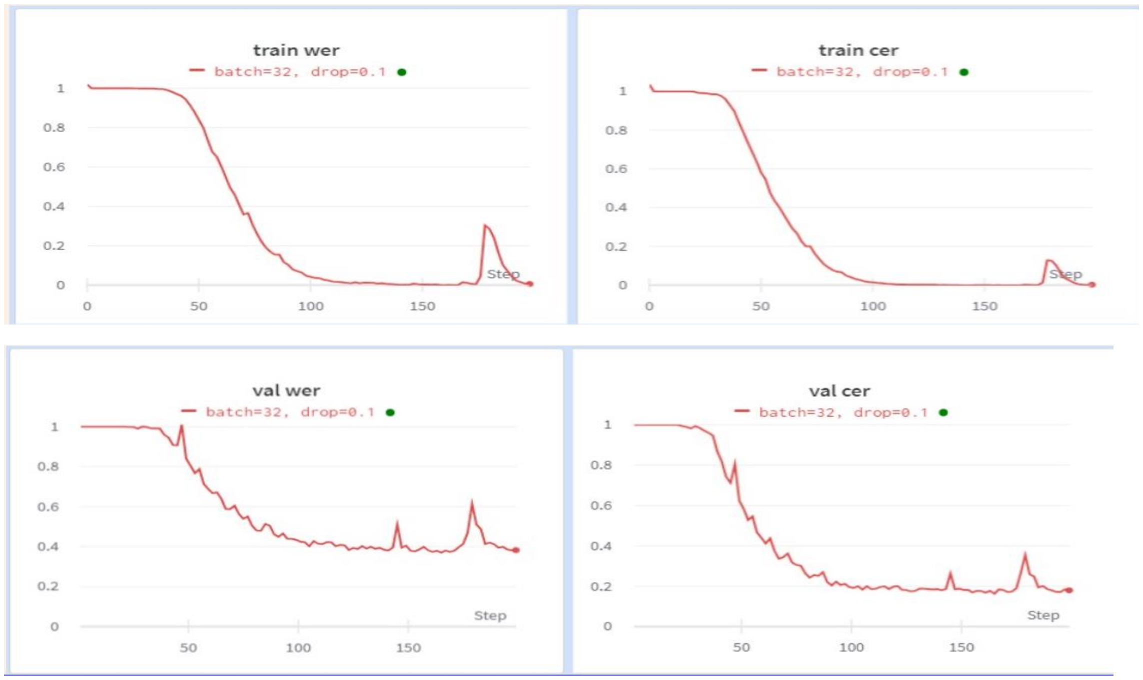
Observing the training WER graph reveals that the model achieved a lower WER during training, potentially indicating overfitting. This could explain why the model’s performance did not improve on unseen data.

<u>Run 3</u>

We decided to use a bigger learning rate then the previous run, to accelerate convergence.

Epoch	Batch Size	Optimizer	Learning Rate	Dropout (after conv)
100	32	Adam	0.001	0.1

Train and Validation Loss:



test wer: 0.315

test cer: 0.141

This result was exciting compared to run 1, because we achieved lower WER on the test set. but the peaks in the train and val error were concerning.

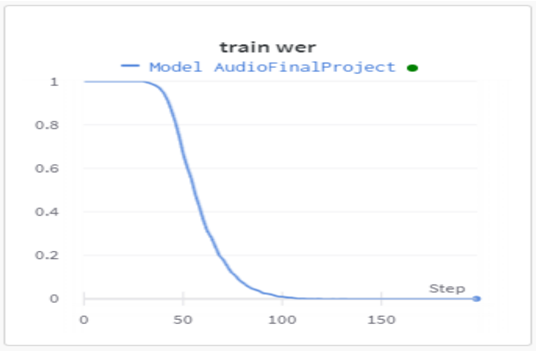
Run 4

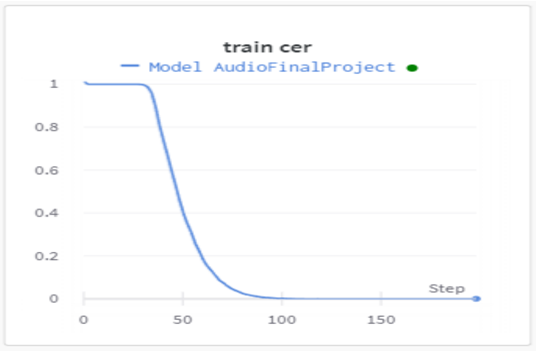
Because of the concerning peaks we decided to stay with the previous learning rate and we decided to change the architecture of the model, we used 4 convolution layers instead of 2.

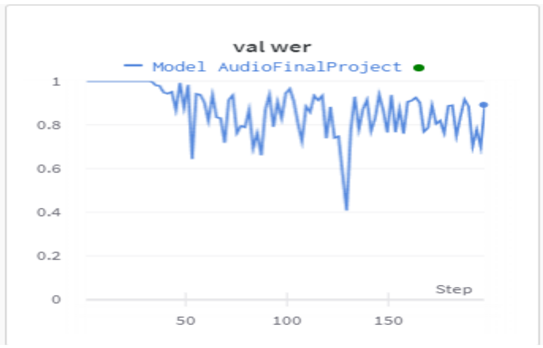
Configuration:

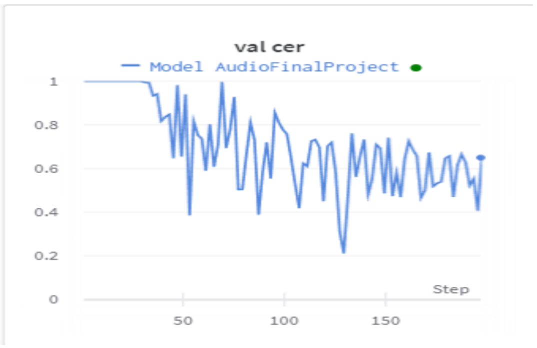
Epoch	Batch Size	Optimizer	Learning Rate	Dropout (after conv)
100	32	Adam	0.0005	0.1

Train and Validation Loss:









test wer: 0.892

test cer: 0.608

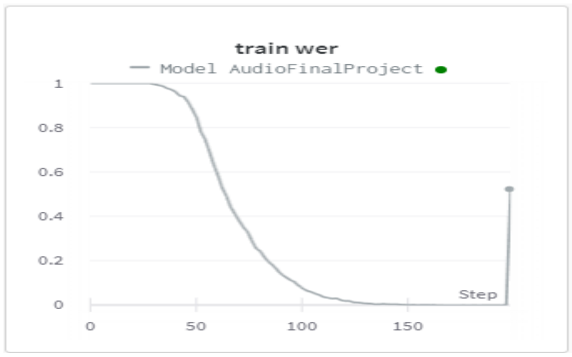
Although the model achieved better WER and CER loss on the train set the loss on the validation and the test set was really bad which indicates that the model suffered from overfit.

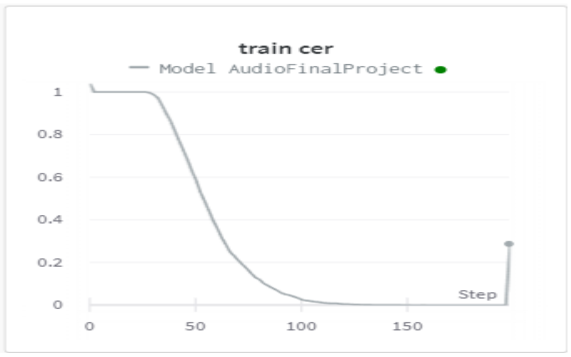
<u>Run 5</u>

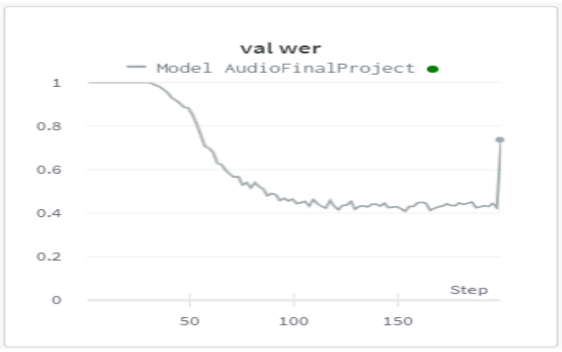
After seeing that adding convolution layers doesn't help the model we decided to use the previous model (2 conv layers), and we started exploring the dropout affect – so we decided not to use dropout.

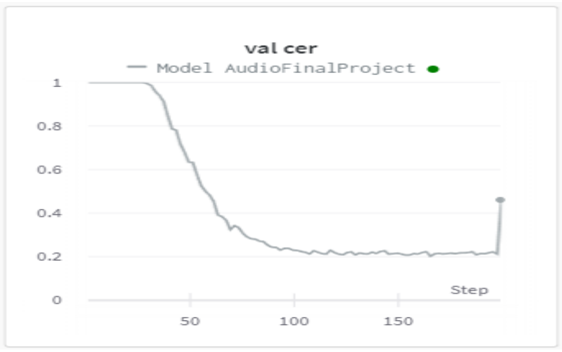
Epoch	Batch Size	Optimizer	Learning Rate	Dropout (after conv)
100	32	Adam	0.001	NO DROPOUT

Train and Validation Loss:









test wer: 0.990

test cer: 0.983

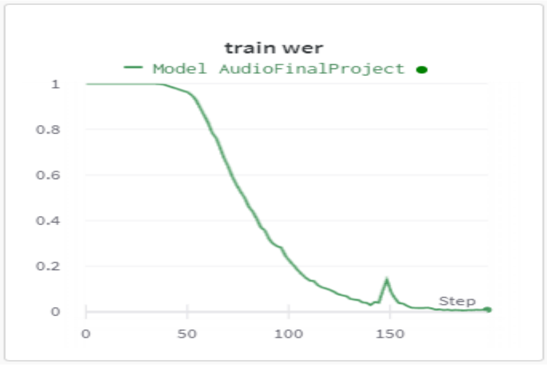
Despite the significant slope towards the end, resulting in dramatic weight changes for the model, it is evident that the previous run exhibited better performance. This outcome suggests that dropout was effective against overfitting, allowing the model to maintain better generalization capabilities.

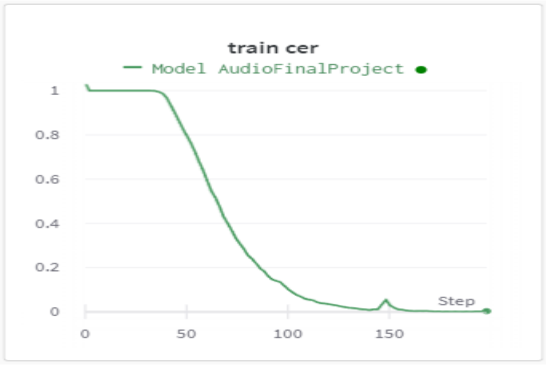
Run 6

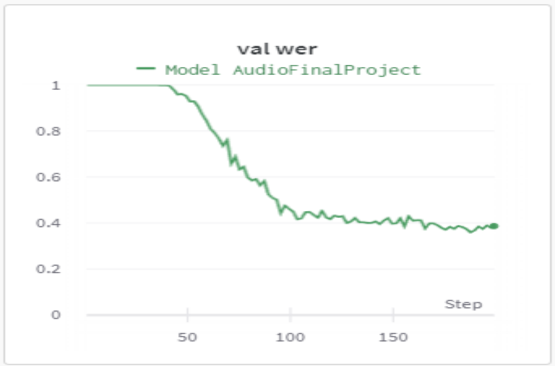
From the previous run we concluded that using dropout layers help and prevents overfit, so we decided to explore the affect of using two layers of dropout, after each convolution.

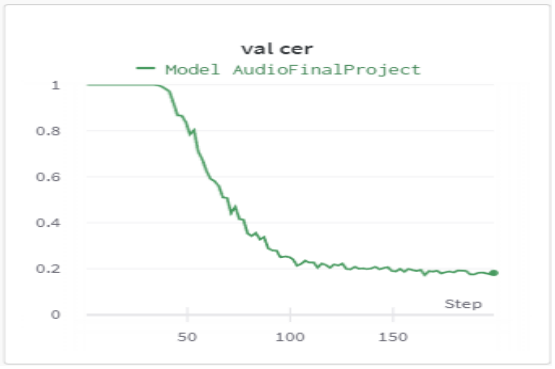
Epoch	Batch Size	Optimizer	Learning Rate	Dropout (after conv)
100	32	Adam	0.001	0.1

Train and Validation Loss:









test wer: 0.376

test cer: 0.184

While this adjustment led to a reduction in validation error, it unfortunately resulted in degradation of test error performance.

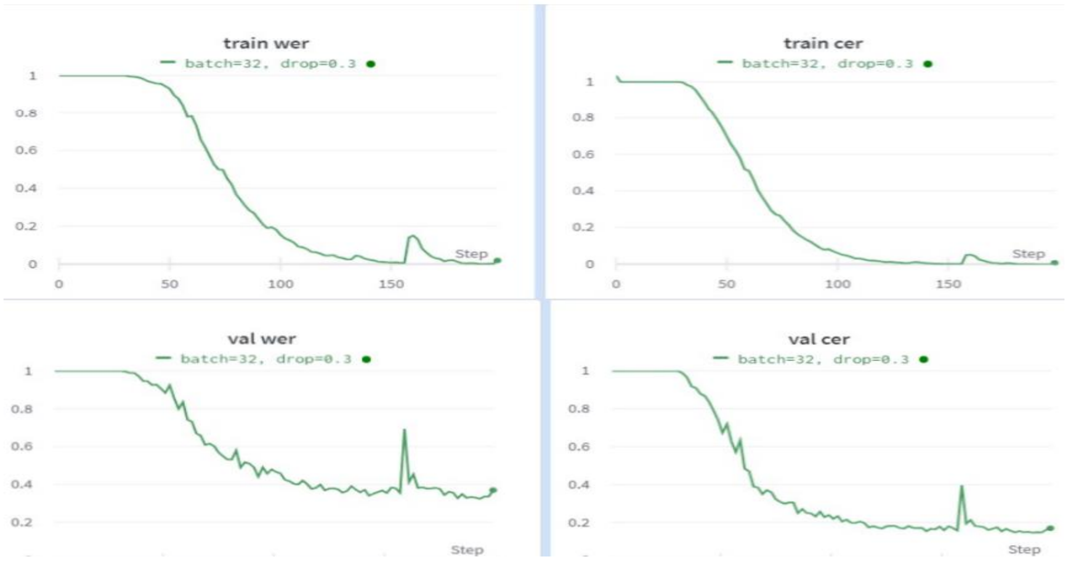
Run 7

After deciding to use one layer of dropout at the end of the convolution network we decided to explore dropout value affect.

Configuration:

Epoch	Batch Size	Optimizer	Learning Rate	Dropout (after conv)
100	32	Adam	0.001	0.3

Train and Validation Loss:



test wer: 0.344

test cer: 0.146

The change seems to have no significant effect (a bit worse – peaks).

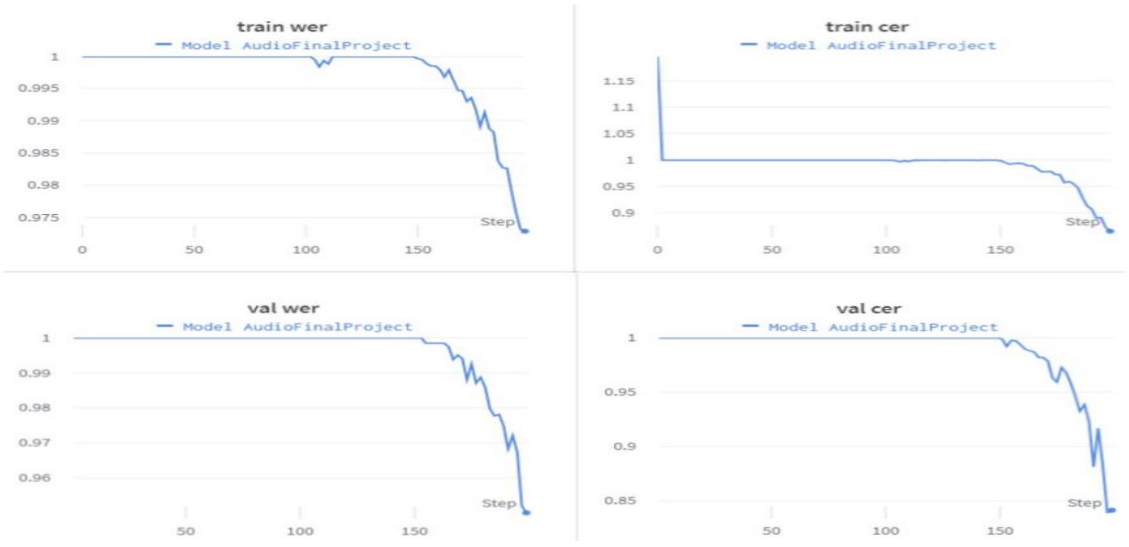
Run 8

Then we wanted to investigate the impact of increasing the batch size (from 32 to 256)

Configuration:

Epoch	Batch Size	Optimizer	Learning Rate	Dropout (after conv)
100	256	Adam	0.001	0.1

Train and Validation Loss:



test wer: 0.966

test cer: 0.860

we found out that this adjustment failed to yield positive results, instead we got larger WER.

Run 9

After exploring different hyperparameters and different number of convolution layers we decided to apply several augmentations on the MelSpectrograms we extracted from the wav files.

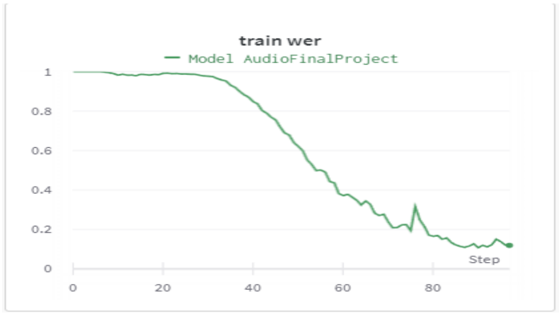
Using torchaudio.transforms.FrequencyMasking(freq_mask_param=?)

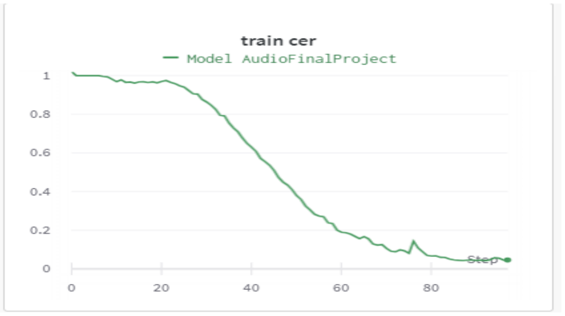
Using torchaudio.transforms.TimeMasking(time_mask_param=?)

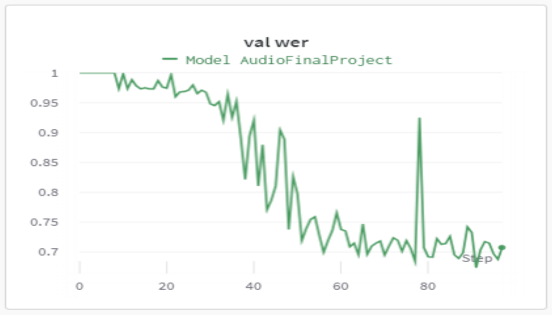
Configuration:

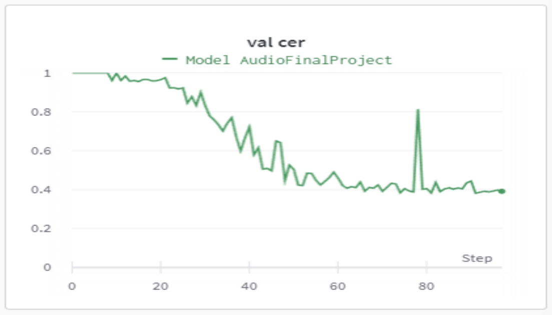
Epoch	Batch Size	Optimizer	Learning Rate	Dropout (after conv)	Freq masking	Time masking
100	32	Adam	0.001	0.1	50	100

Train and Validation Loss:









Test wer: 0.581

Test cer: 0.324

At this moment we didn't understand why data augmentation didn't help to generalize.

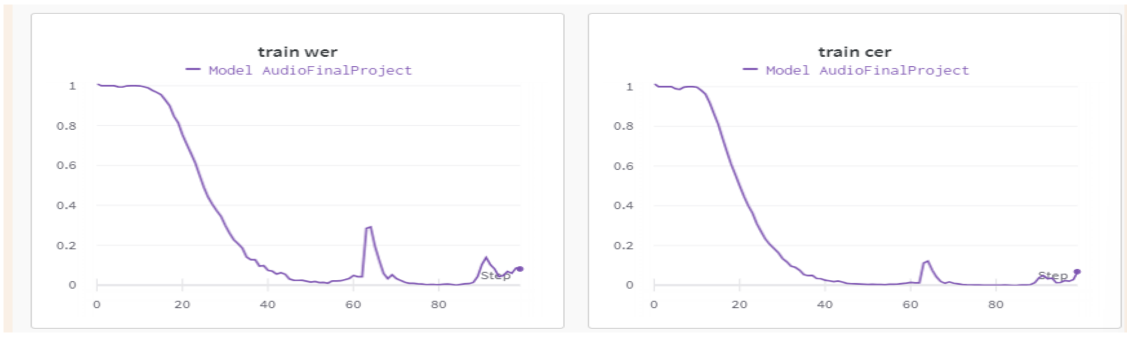
<u>Run 10</u>

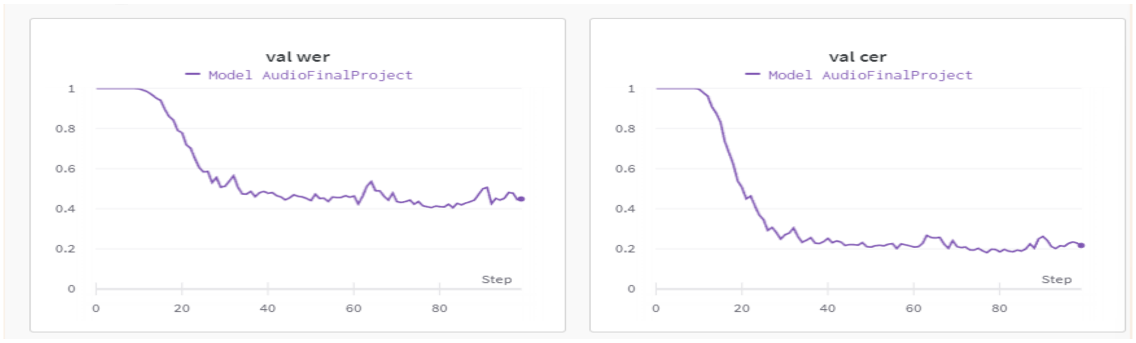
We found masking and data augmentation very interesting, and we wanted to explore further, so we used smaller values of masking on the frequency and time.

Configuration:

Epoch	Batch Size	Optimizer	Learning Rate	Dropout (after conv)	Freq masking	Time masking
100	32	Adam	0.001	0.1	15	35

Train and Validation Loss:





test wer: 0.443

test cer: 0.385

using smaller values for both time and frequency masking on the spectrogram has proven to improve results.

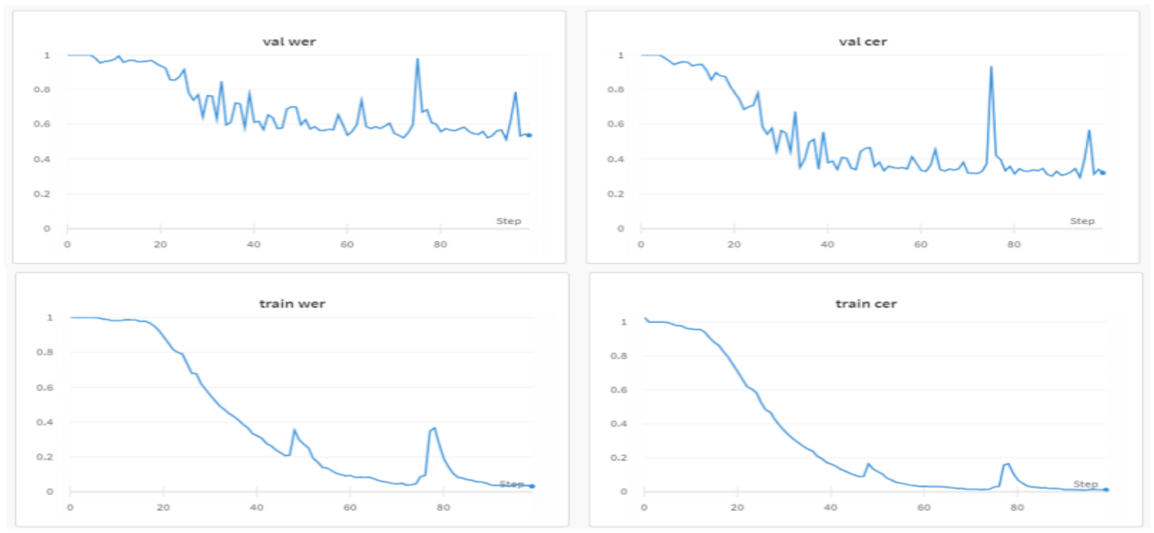
Although we thought masking might prevent overfit and help the model to generalize – which will be notice in lower test WER it didn't improved. Our suspicion is that because we train the model on a small data set (853 training wavs) masking didn't help, and we think that using masking (augmentation) can help when given bigger train set.

Run 11

After reading about time shift we decided to try to implement it without any further augmentations as before, this might help the model extract the features in a clearer way

Epoch	Batch Size	Optimizer	Learning Rate	Dropout (after conv)	time shift	Freq masking	Time masking
100	32	Adam	0.001	0.1	0.4	X	X

Train and Validation Loss:



test wer: 0.581

test cer: 0.324

Although we though shifting the wav file might help the model to generalize, maybe the use of fixed shift value was limiting, or that we shifted the hole dataset and not only some of it.

Phase 3:

Next, we understood that our model from the previous phase started to remind us of the DeepSpeech model we saw in class, so we decided to implement a DeepSpeech model.

The changes that needed to be done were:

After 1 layer of convolution we used:

ResidualCnn – uses skip connection, layer normalization (normalizing the activations across the features of a single data point), relu activation function and dropout after each layer.

BidirectionalGRU – instead of using bidirectional LSTM we used GRU (which is faster to use and saved us a lot of time when using google colab) because our model got deep and complex.

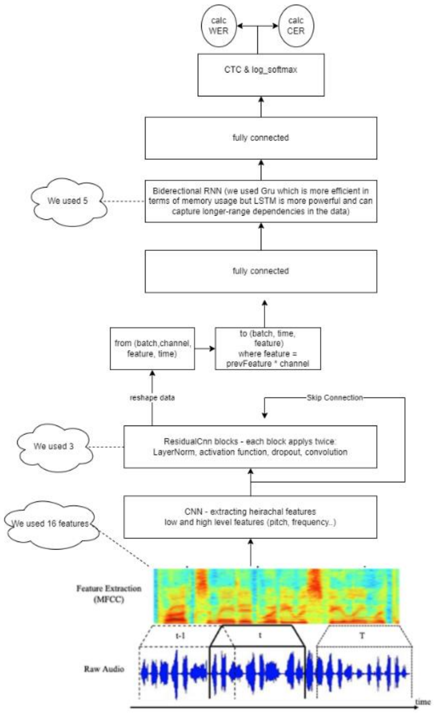
Another change was that we extracted from the wav files MFCC with 16 features (represent the spectral characteristics of audio signals in a way that is more suitable for ML) and not MelSpectrogram (shown in run 1 and 2)

In this part we used different configurations.

Conclusion from this phase:

Run number 11 achieved the best WER – 15%.

PipeLine


calc WER
calc CER
CTC & log_softmax
fully connected
We used 5
Biderectional RNN (we used Gru which is more efficient in terms of memory usage but LSTM is more powerful and can capture longer-range dependencies in the data)
fully connected
from (batch,channel, feature, time)
to (batch, time, feature) where feature = prevFeature * channel
reshape data
Skip Connection
We used 3
ResidualCnn blocks - each block applys twice: LayerNorm, activation function, dropout, convolution
We used 16 features
CNN - extracting heirachal features low and high level features (pitch, frequency..)
Feature Extraction (MFCC)
t-1
t
T
Raw Audio
time

<u>Run 1</u>

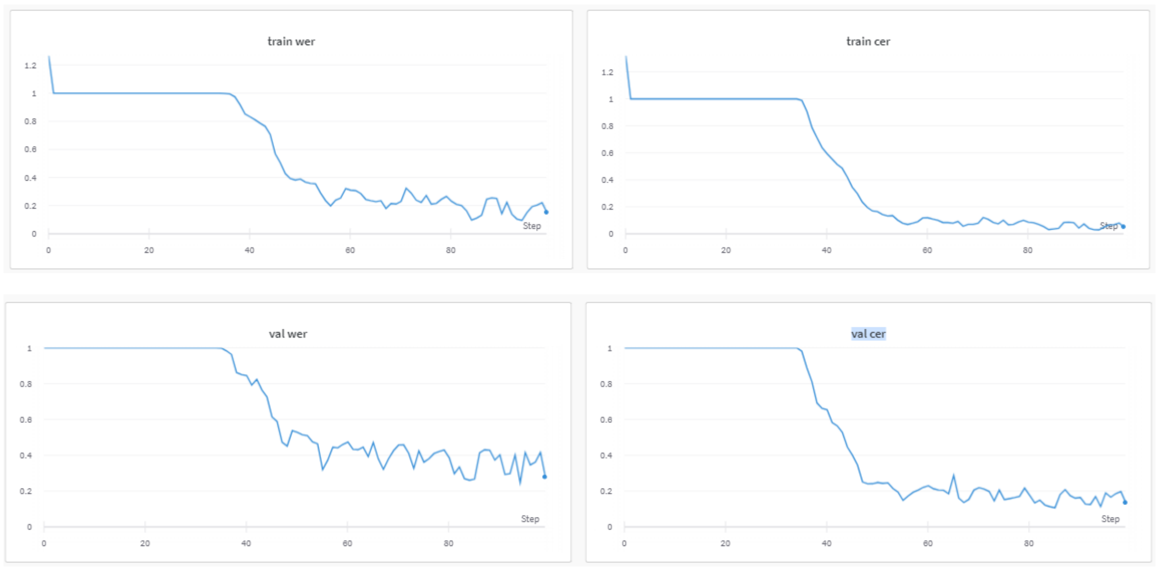
First we wanted to compare the results between the use of MelSpectrogram to MFCC.

We used MelSpectrogram as we did in previous runs.

Configuration:

Epoch	Batch Size	Optimizer	Learning Rate	Dropout (after conv)
100	20	Adam	0.0005	0.1

Train and Validation Loss:



Test wer: 0.299

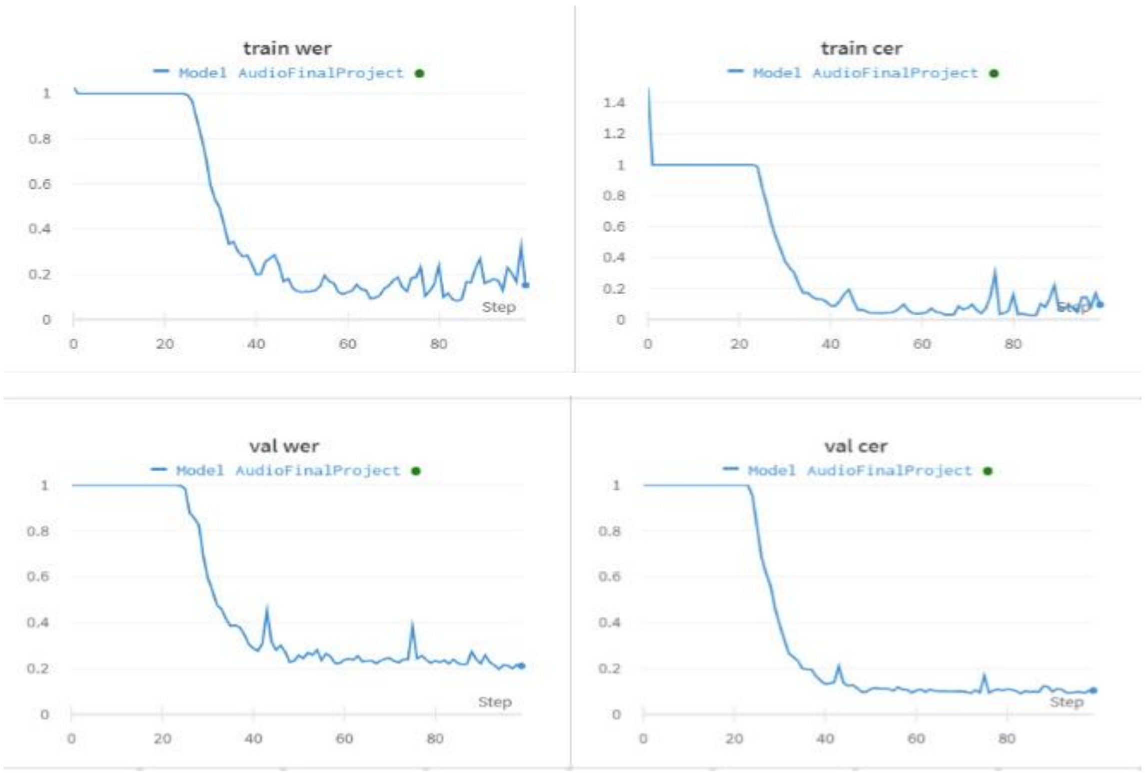
Test cer: 0.131

By using the new model we managed to improve the WER to 29% compared to previous WER of 31%.

Run 2

Then we run the same model with the same configurations except that we used MFCC instead of MelSpectrogram.

Train and Validation Loss:



test wer: 0.248

test cer: 0.085

MFCCs are a more compact representation of the sound than mel-spectrograms, they are often used in tasks where classification accuracy is more important than interpretability. This is why when we used MFCC our test WER improved.

From now on we used MFCC.

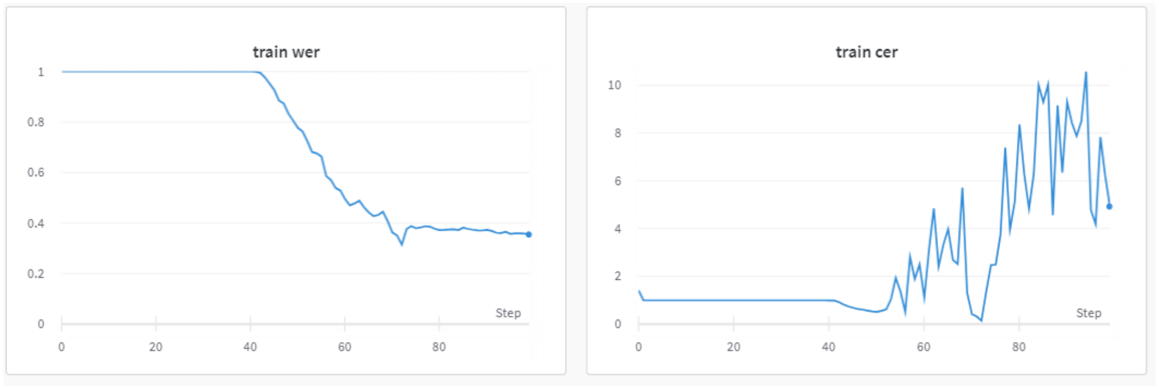
Run 3

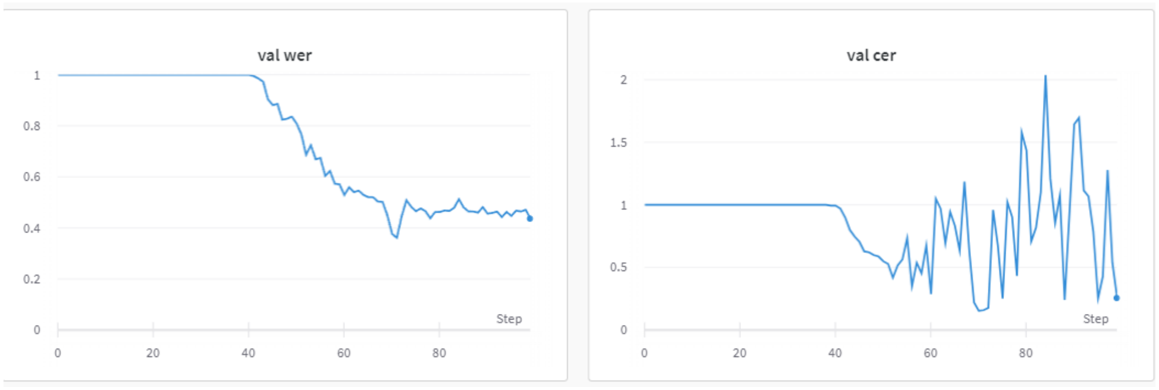
Then we wanted to check the same run but with adamW Optimizer (prevents the weights from becoming too large, which can help to improve the generalization performance of the model).

Configuration:

Epoch	Batch Size	Optimizer	Learning Rate	Dropout (after conv)
100	20	AdamW	0.0005	0.1

Train and Validation Loss:





Test wer:0.432

Test cer:0.279

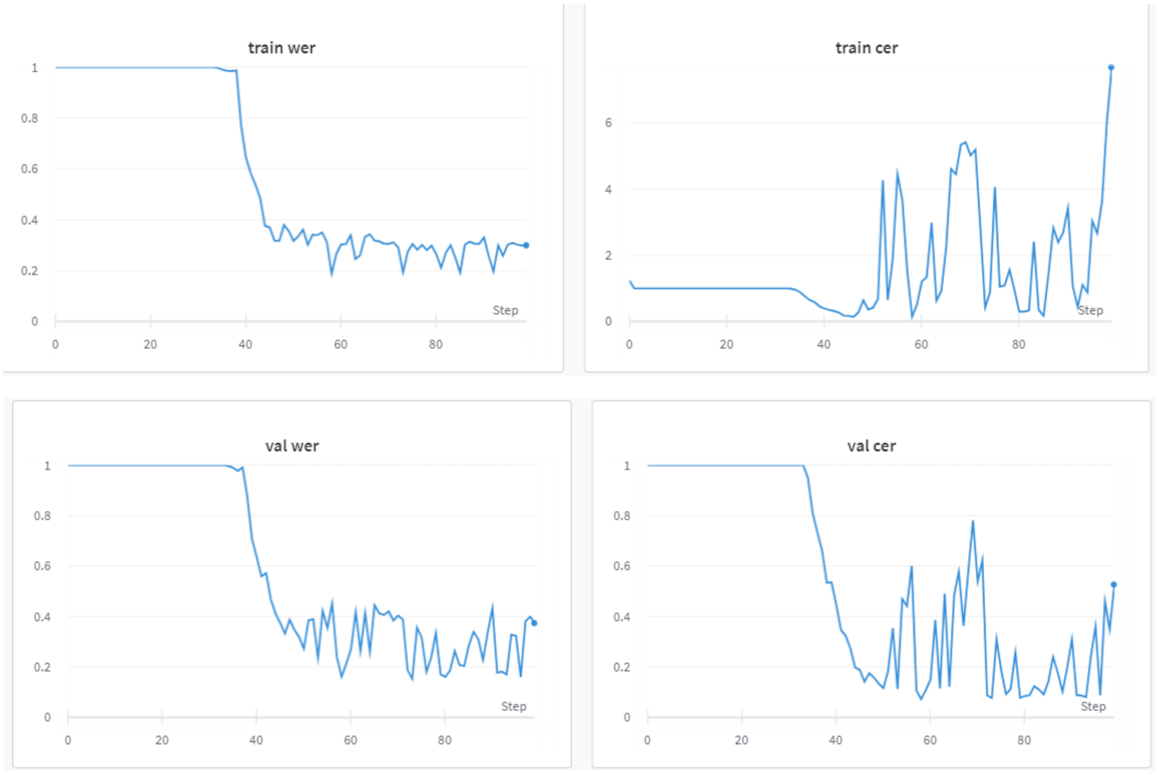
Although the graph looked better (more smooth) we got bigger test loss.

<u>Run 4</u>

After the results we got in the previous run we decided to continue with Adam Optimizer and use the same configurations.

Then we tried to deeper the architecture of the model , we used 4 residualCNN and 6 GRU.

Train and Validation Loss:



Test wer: 0.269

Test cer: 0.118

Although the test WER wasn’t that high, the graph contained oscillations.

From now until run 8, we used the deeper network.

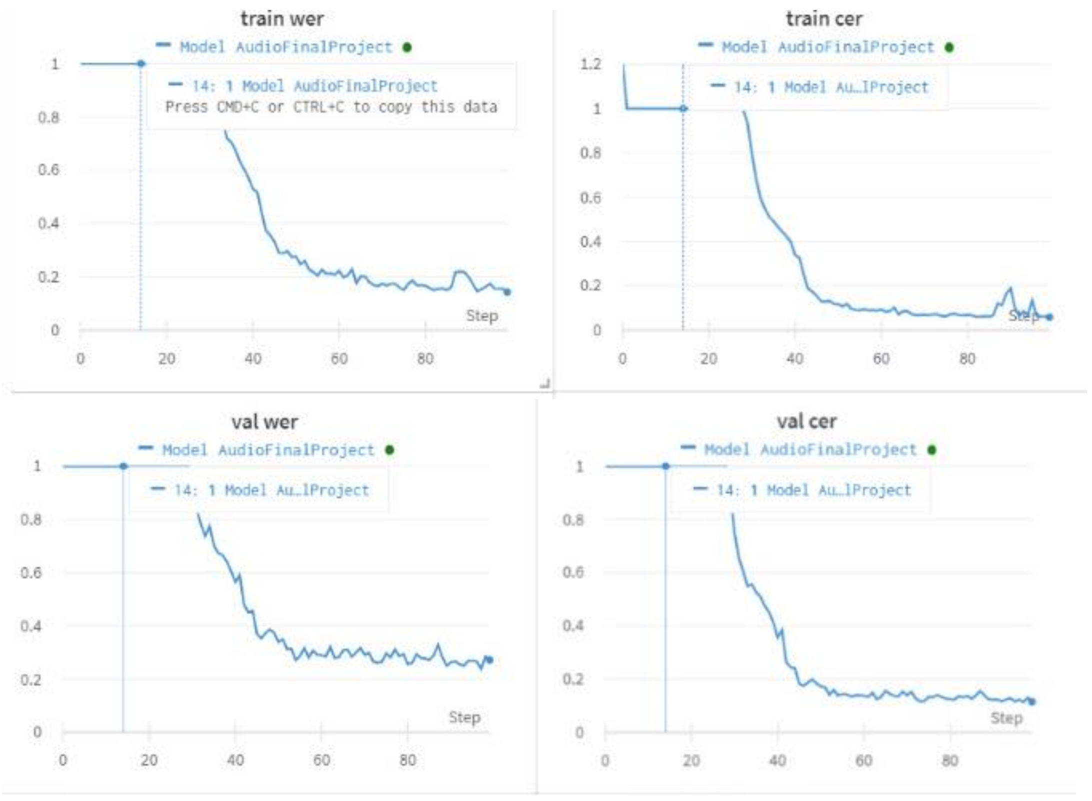
Run 5

Then we wanted to check the effect of using smaller batch size (8 compare to 20)

Configuration:

Epoch	Batch Size	Optimizer	Learning Rate	Dropout (after conv)
100	8	Adam	0.0005	0.1

Train and Validation Loss:



test wer: 0.235959435863 28197

test cer: 0.10147772251502427

smaller batch sizes introduce more randomness into the training process. Each batch represents a smaller subset of the training data, leading to noisier estimate real loss. This helps the model to escape local minima and explore the weight space more effectively.

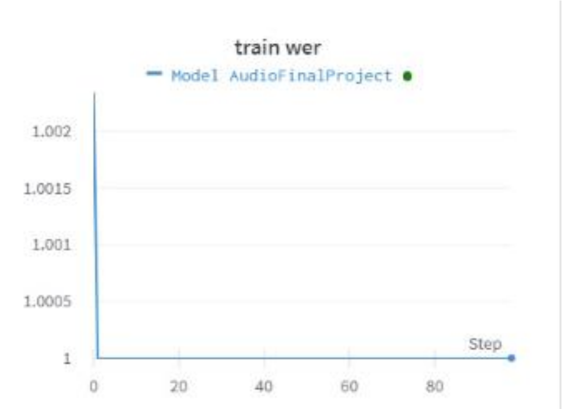
Run 6

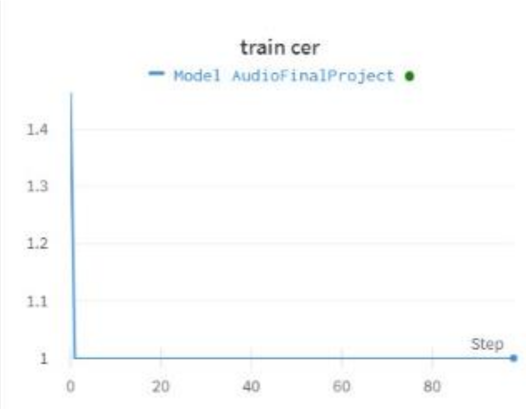
After checking the affect of smaller batch size we wanted to explore the option of using bigger learning rate (0.001 compared to 0.0005)

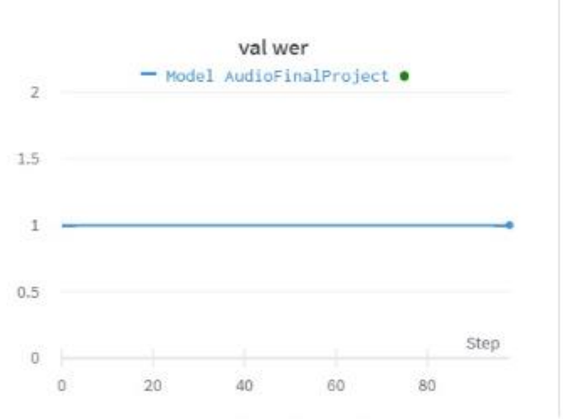
Configuration:

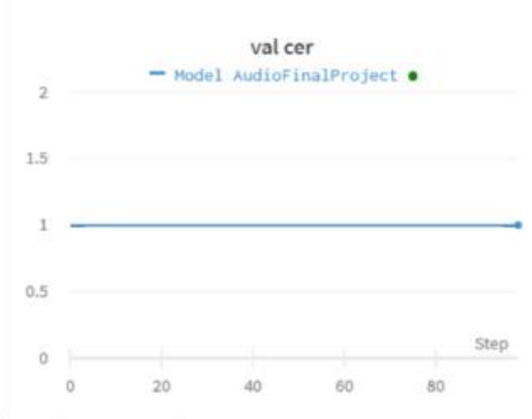
Epoch	Batch Size	Optimizer	Learning Rate	Dropout (after conv)
100	20	Adam	0.001	0.1

Train and Validation Loss:









test wer: 1.0

test cer: 1.0

Increasing the learning rate can speed up the convergence process but it didn’t help, the result was really bad, the reason might be that the learning rate was to big and it caused the network to take large steps in the weight space without effectively following the gradient to reach local minimum.

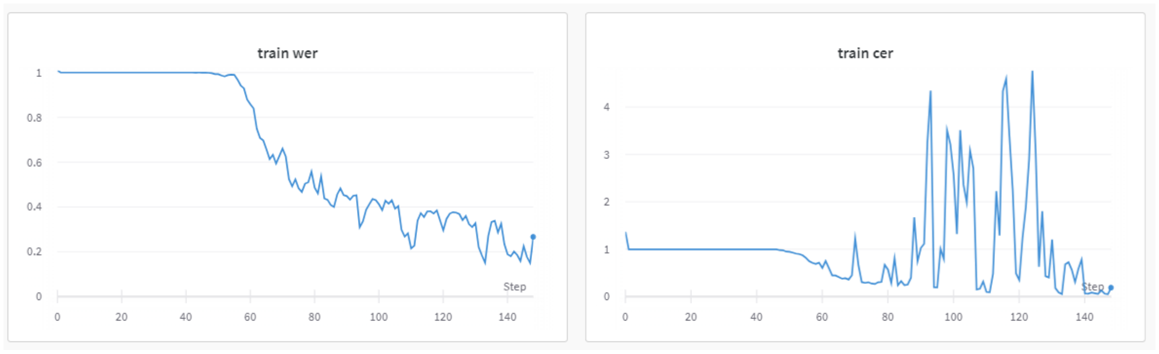
<u>Run 7</u>

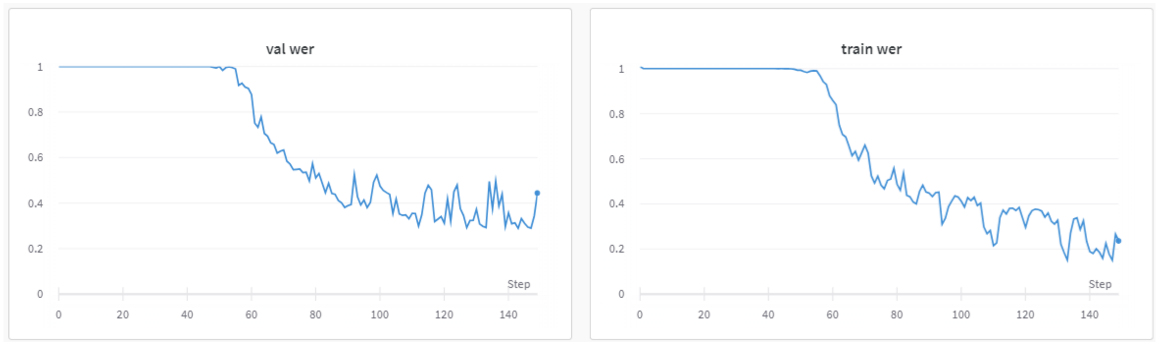
Then we wanted to check the model performance using more epochs with smaller batch size.

Configuration:

Epoch	Batch Size	Optimizer	Learning Rate	Dropout (after conv)
150	8	Adam	0.0005	0.1

Train and Validation Loss:





test wer: 0.433
test cer: 0.2

Notice that the test error is calculated after training the model, in the last epoch we can notice that there is a peak in the error of the validation and train sets, but if we would train the same model with more epochs we might get loss that doesn't change that much,

Run 8

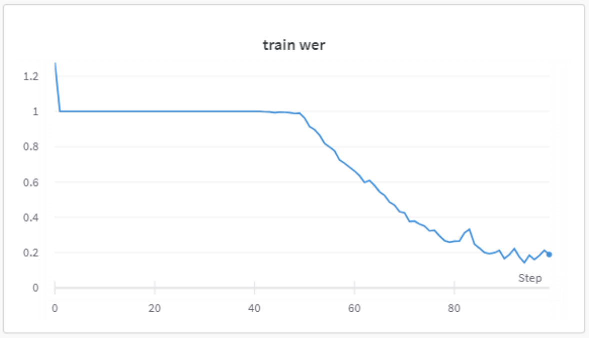
Then we got back to the deep speech original model that used 3 ResidualCnn and 5 GRU biderectional because training deep nn takes a lot of time.

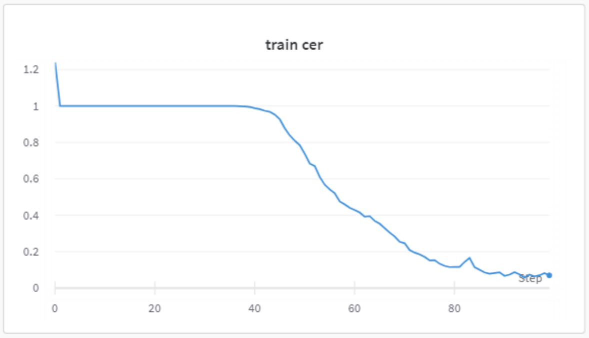
In phase 2 we checked our model with Time and Frequency Masking, we wanted to check our DeepSpeech model with masking but this time we didn't want to mask the whole training set, we randomly masked the train set each epoch.

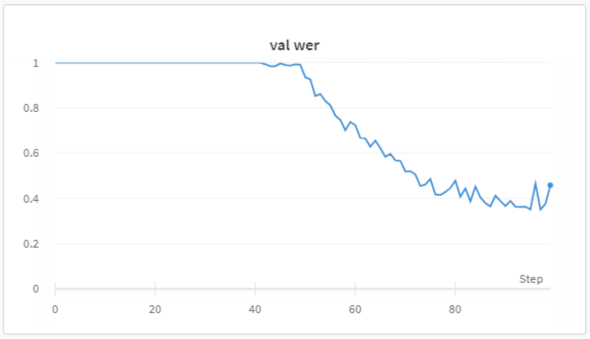
Configuration:

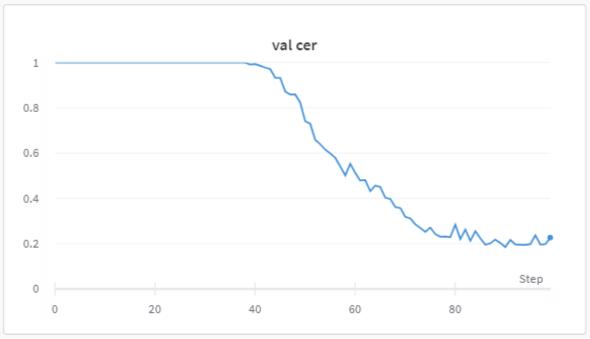
Epoch	Batch Size	Optimizer	Learning Rate	Dropout (after conv)	Time Masking	Freq Masking
100	8	Adam	0.0005	0.1	20	10

Train and Validation Loss:









test wer: 0.413
test cer: 0.2

notice that we applied the augmentation when loading the data, which means that if augmentation_prob > 0.5 (as implemented in our code) on each epoch the model will see the same augmentation_prob and he wont encounter the non augmented data which is very problematic.

Run 9

In our code we used CustumDataset which takes several parameters during initialization and saves them, this class implemented the getItem function responsible for fetching individual sample when provided index.

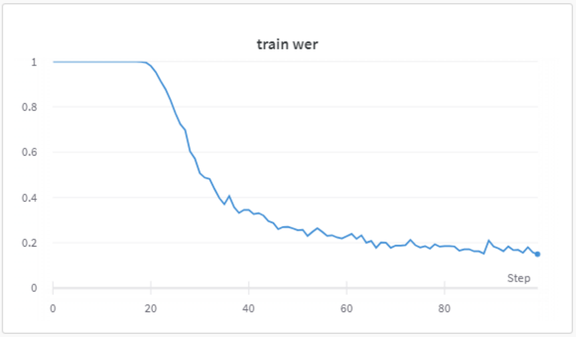
The instance of this class is sent to DataLoader, each epoch DataLoader splits the data to different batches, each data in this batch (if its in train mode) might get augmented, this will increase robustness and because the model will be exposed to a wider range of speech patterns he will be able to generalize better, and it will prevent overfit.

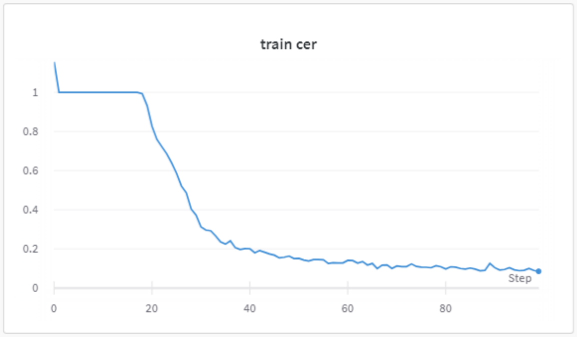
Another change is that we used 4 layers of cnn, and 3 layers of rnn, and bigger batch size.

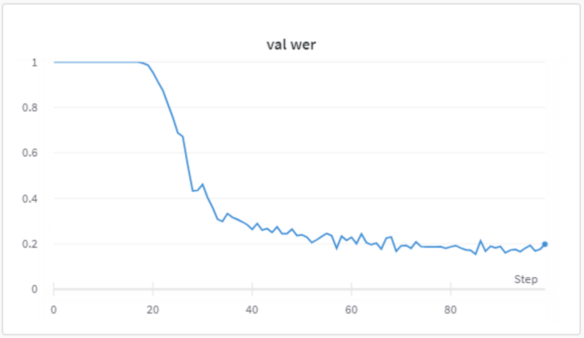
Configuration:

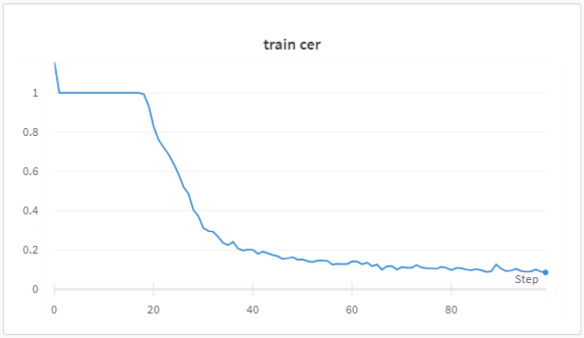
Epoch	Batch Size	Optimizer	Learning Rate	Dropout (after conv)	Masking prob	Time Masking	Freq Masking
100	16	Adam	0.0005	0.1	0.4	20	10

Train and Validation Loss:









test wer: 0.181
test cer: 0.071

By combining augmentation techniques with changes in the model's architecture and batch size, the model's performance was significantly improved. Notably, the model began learning earlier reaching meaningful progress around epoch 20 compared to previous models that required more time (around epochs 40-50) to begin improving. The loss curve exhibited smother trend with fewer spikes, and ultimately the WER achieved on the test set was reduced to 18%!

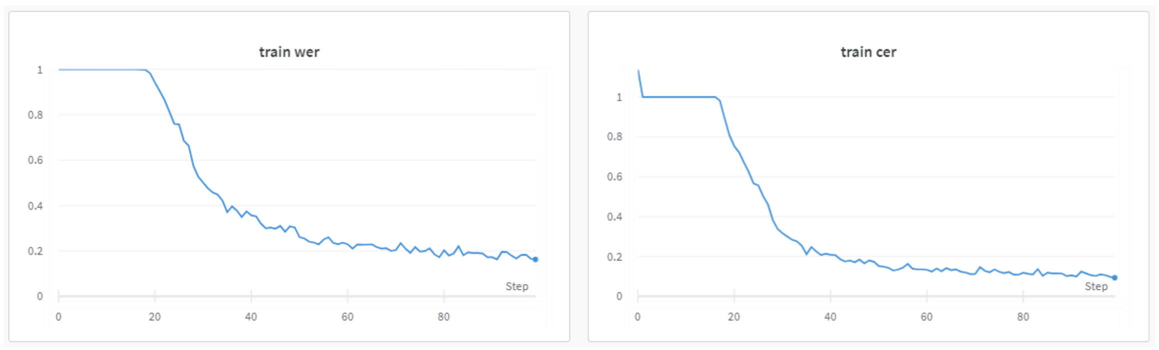
<u>Run 10</u>

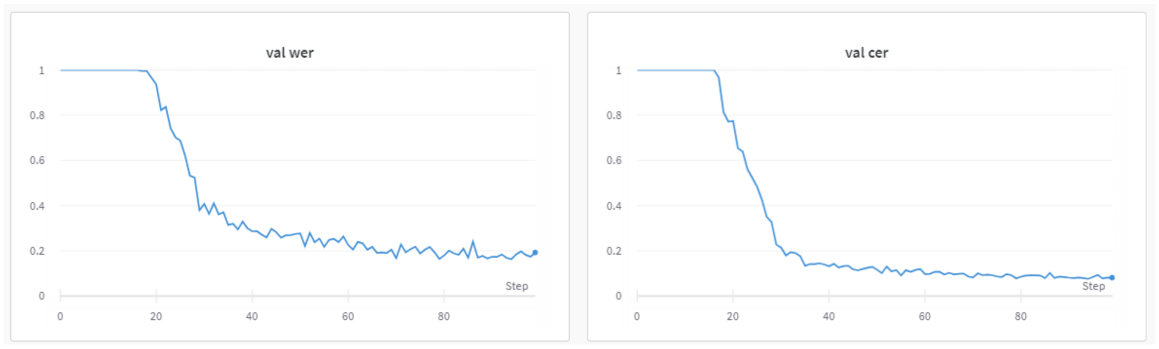
After achieving 18% WER we decided to check the batch size affect again, using smaller batch size.

Configuration:

Epoch	Batch Size	Optimizer	Learning Rate	Dropout (after conv)	Masking prob	Time Masking	Freq Masking
100	8	Adam	0.0005	0.1	0.4	20	10

Train and Validation Loss:





test wer: 0.206
test cer: 0.092

Using smaller batch size might introduce to much noise so we decided to use the previous configuration (16 batch).

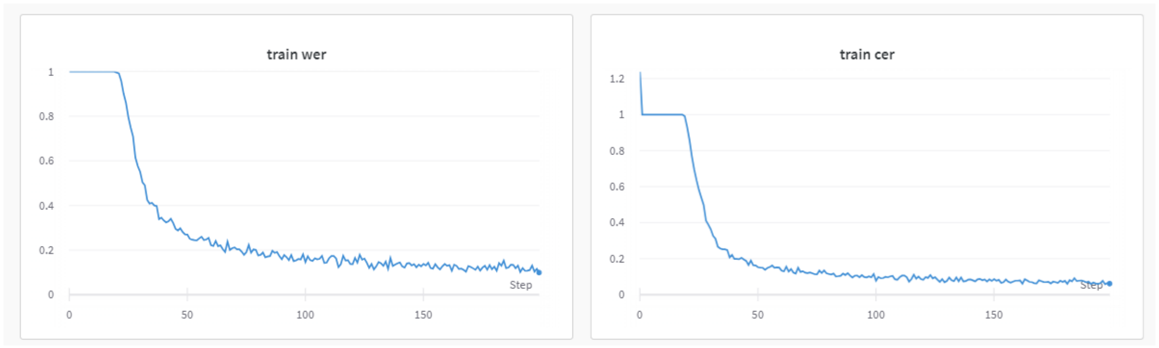
<u>Run 11</u>

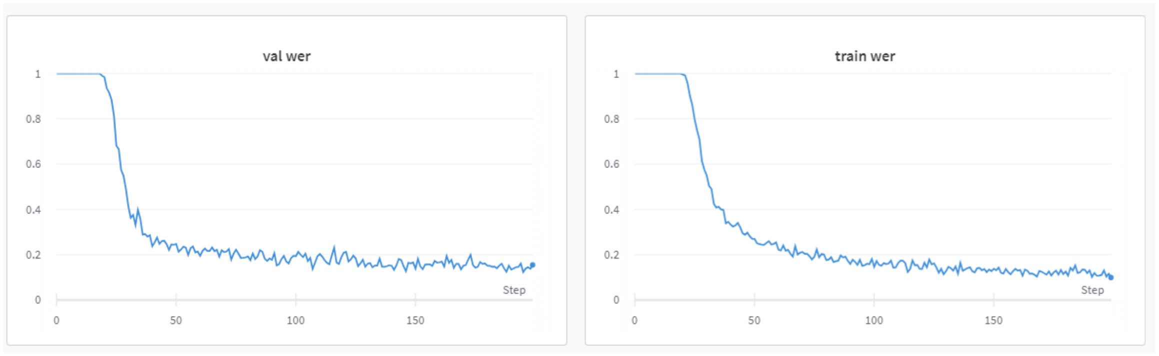
In our last configuration we run the model with 200 epochs instead of 100

Configuration:

Epoch	Batch Size	Optimizer	Learning Rate	Dropout (after conv)	Masking prob	Time Masking	Freq Masking
200	16	Adam	0.0005	0.1	0.4	20	10

Train and Validation Loss:





test wer: 0.152
test cer: 0.0569

This was our best result.