Speech Recognition
Comparison of Neural Networks

CSIT687 Masters Project– Spring 2020

Litha Thampan

Project Phases

DATA PREPARATION

Identify a good data set for speech recognition. Prepare the data for better feature extractions.

FEATURE IDENTIFICATION

Identify good feature representations for training data. Identify optimization techniques for the features

BUILD MODELS

Build different deep neural network models. Choose between various activation functions and optimizers

TEST MODELS

Test models with prepared data set to evaluate the loss and performance of the models

COMPARE

Compare the training and validation losses and draw conclusions

LibriSpeech Data Stats

Subset	Hours	Per-speaker Minutes	Female Speakers	Male Speakers	Total Speakers
dev-clean	5.4	8	20	20	40
test-clean	5.4	8	20	20	40
dev-other	5.3	10	16	17	33
test-other	5.1	10	17	16	33
train-clean-100	100.6	25	125	126	251
train-clean-360	363.6	25	439	482	921
train-other-500	496.7	30	564	602	1166

LENGTH

1000 hours

of noise free English speech data

SAMPLE RATE

16kHz

Frequency range in the normal conversation spectrum

MALE : FEMALE SPEAKERS

1:1

Ratio of male and female speakers

Environment

Jupyter Notebook
- Interactive environment for python

Keras 2.3.1
- Part of TensorFlow 2
- Allows easy layering of neural networks

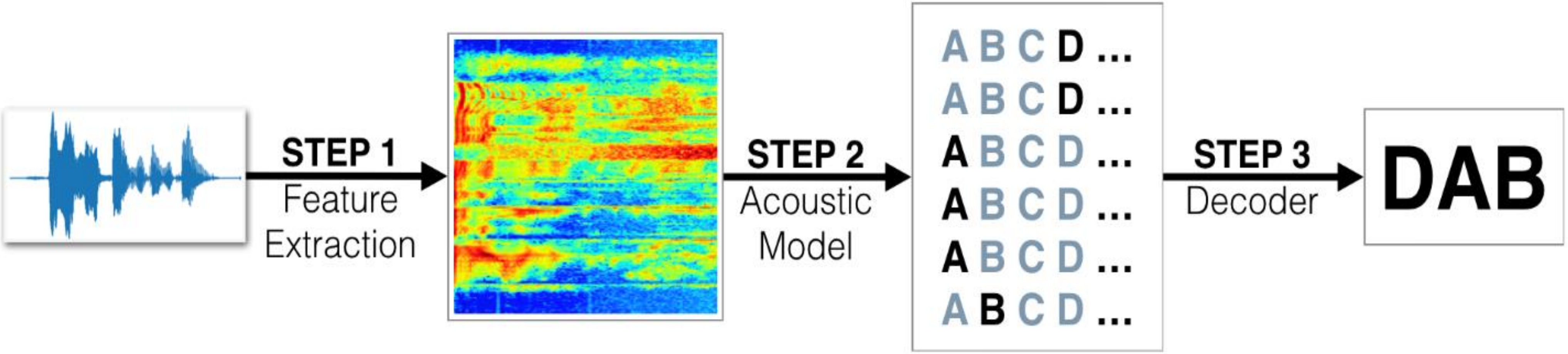
TensorFlow 2.1.0
- Powerful framework for neural network
- Easy integration with GPU

Python 3.6
- Programming language with wrapper modules for several frameworks and visualizations

CUDA 10.1
- NVIDIA GPU Compute Framework

Ubuntu 18.04
- Linux Distribution which supports all major frameworks.

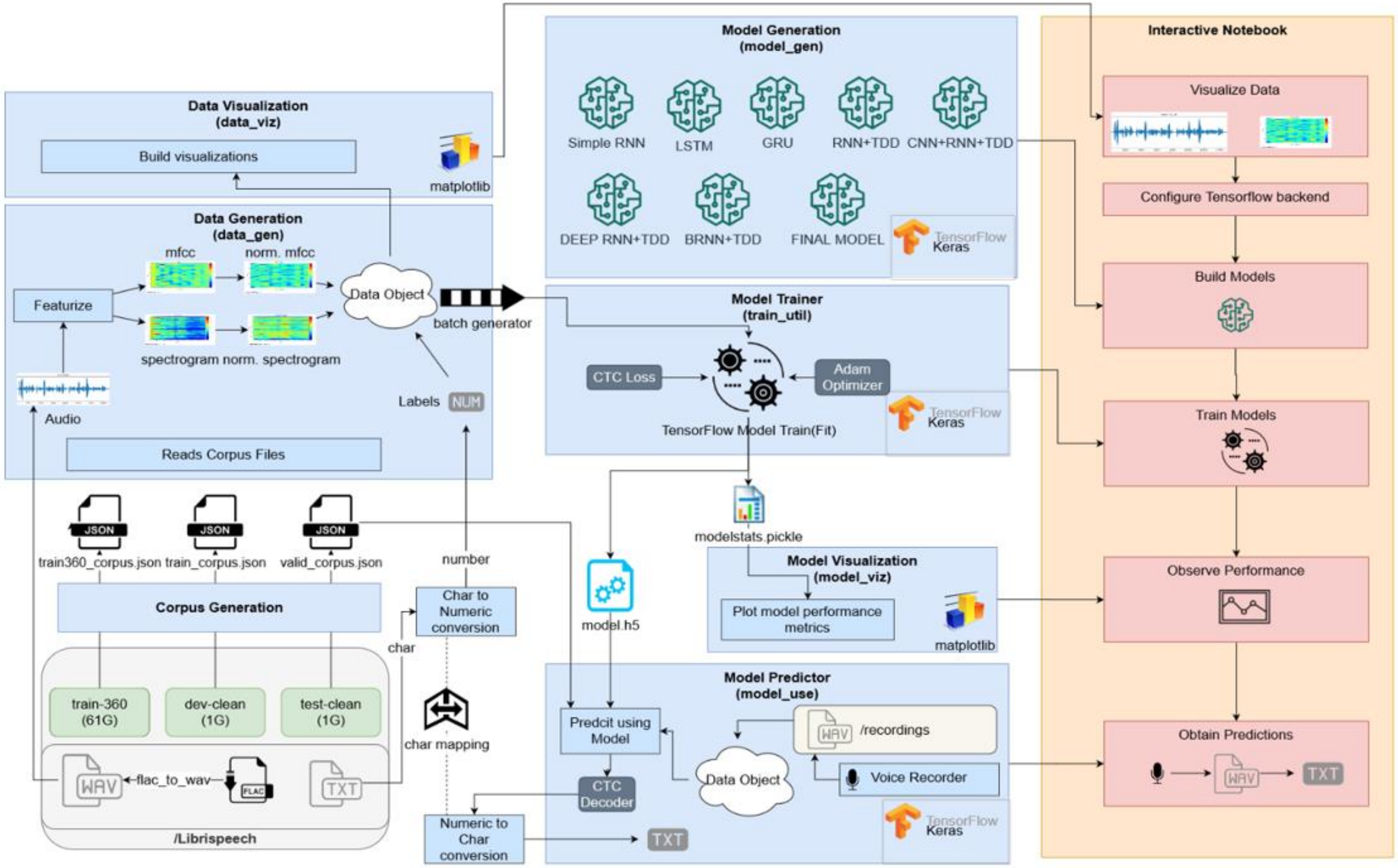
Overall Pipeline



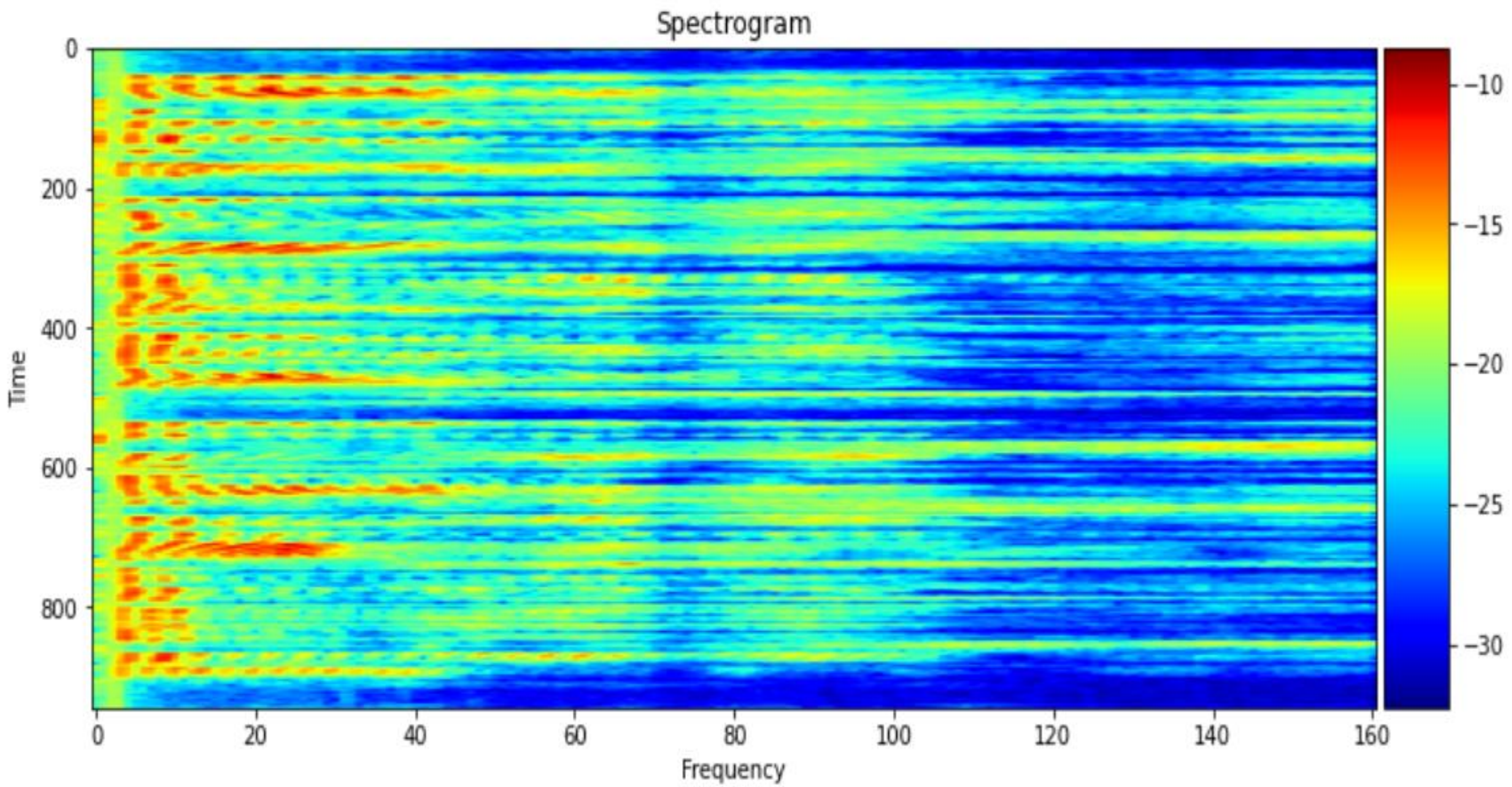
Label Encoding

'	SPACE	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27

Overall Architecture



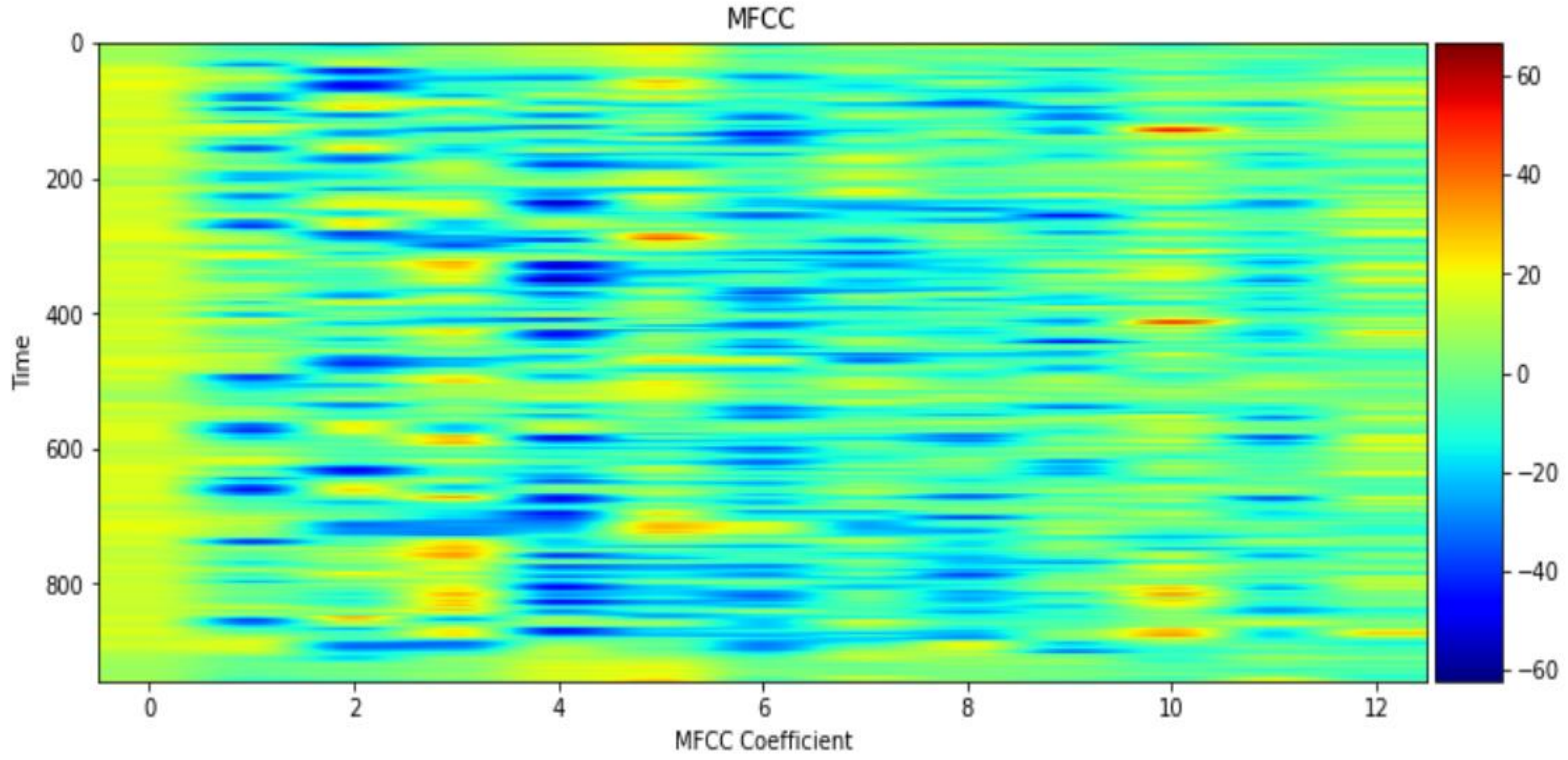
Features – Spectrogram



Shape of Spectrogram : (944, 161)

- A visual representation of the spectrum of frequencies of a signal as it varies with time.
- Represented as a 2D tensor of shape (X,161) where X is the length of the speech.

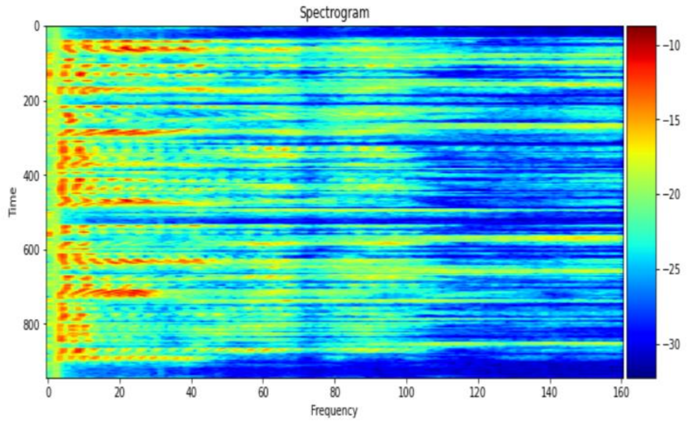
Features – MFCC



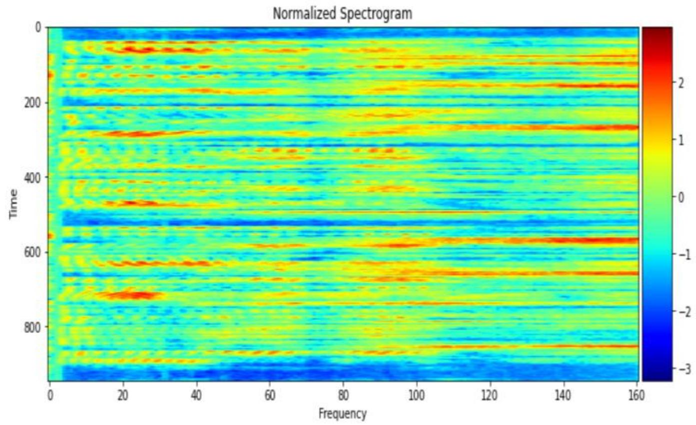
Shape of MFCC : (944, 13)

- Mel-Frequency Cepstral Coefficients of a signal are small set of features which describe the overall shape of a spectral envelope.
- Represented as a 2D tensor of shape (X,13) where X is the length of the speech.

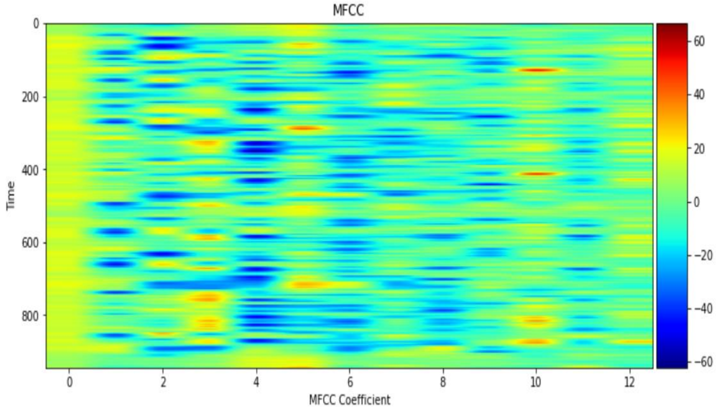
Features – Normalization



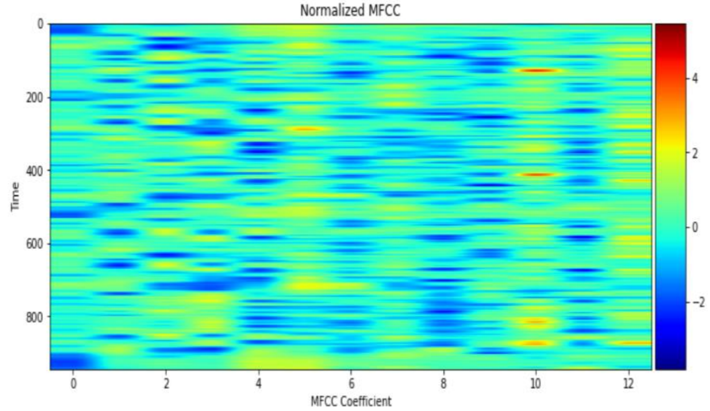
Shape of Spectrogram : (944, 161)



Shape of Spectrogram : (944, 161)



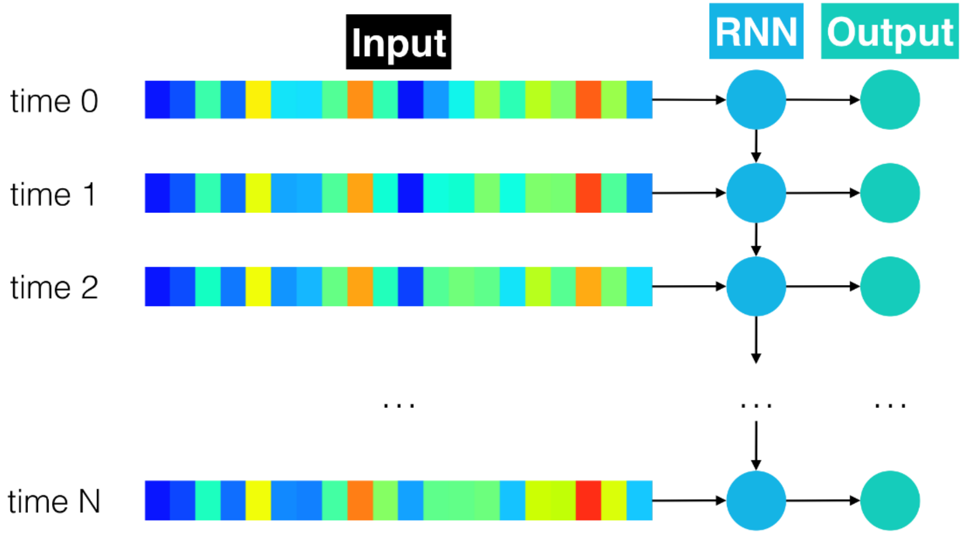
Shape of MFCC : (944, 13)

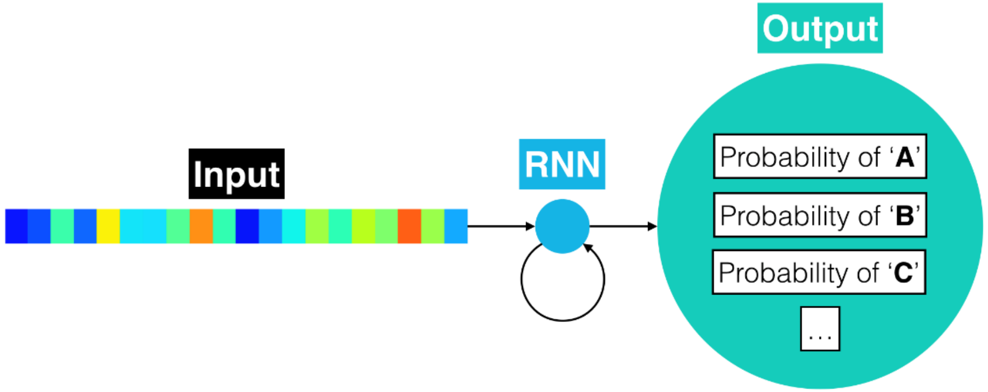


Shape of MFCC : (944, 13)

- Speeds up convergence
- Reduces the impact of noise in the data

Models – Single Layer RNN – Arch





- Simplest RNN Layer where Input is directly fed to RNN
- Implemented on SimpleRNN , LSTM and GRU

Models – Single Layer RNN – Summary

MFCC Models

```
Model: "simple_mfcc"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
the_input (InputLayer)       [(None, None, 13)]        0
_________________________________________________________________
rnn (SimpleRNN)              (None, None, 29)          1247
_________________________________________________________________
softmax (Activation)         (None, None, 29)          0
=================================================================
Total params: 1,247
Trainable params: 1,247
Non-trainable params: 0
_________________________________________________________________
```

```
Model: "lstm_mfcc"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
the_input (InputLayer)       [(None, None, 13)]        0
_________________________________________________________________
rnn (LSTM)                   (None, None, 29)          4988
_________________________________________________________________
softmax (Activation)         (None, None, 29)          0
=================================================================
Total params: 4,988
Trainable params: 4,988
Non-trainable params: 0
_________________________________________________________________
```

```
Model: "gru_mfcc"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
the_input (InputLayer)       [(None, None, 13)]        0
_________________________________________________________________
rnn (GRU)                    (None, None, 29)          3828
_________________________________________________________________
softmax (Activation)         (None, None, 29)          0
=================================================================
Total params: 3,828
Trainable params: 3,828
Non-trainable params: 0
_________________________________________________________________
```

Models – Single Layer RNN – Summary

Spectrogram Models

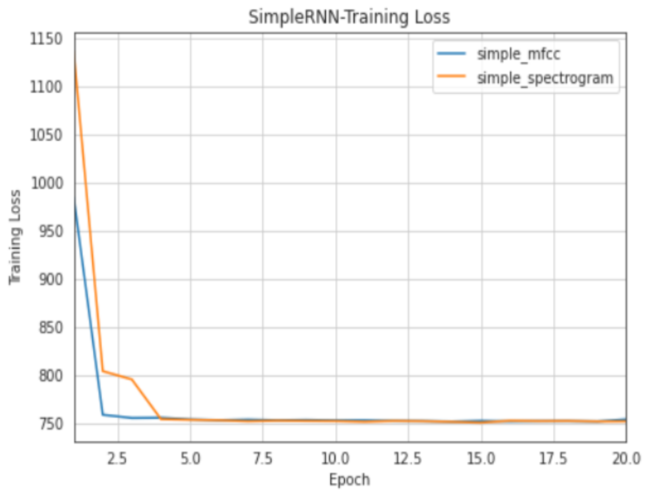
```
Model: "simple_spectrogram"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
the_input (InputLayer)       [(None, None, 161)]       0
_________________________________________________________________
rnn (SimpleRNN)              (None, None, 29)          5539
_________________________________________________________________
softmax (Activation)         (None, None, 29)          0
=================================================================
Total params: 5,539
Trainable params: 5,539
Non-trainable params: 0
_________________________________________________________________
```

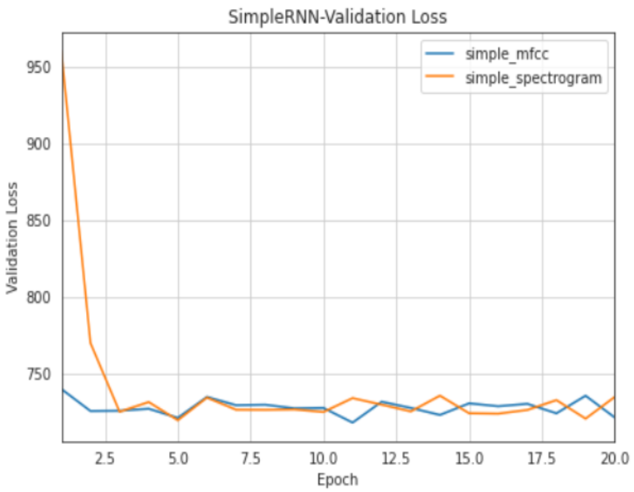
```
Model: "lstm_spectrogram"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
the_input (InputLayer)       [(None, None, 161)]       0
_________________________________________________________________
rnn (LSTM)                   (None, None, 29)          22156
_________________________________________________________________
softmax (Activation)         (None, None, 29)          0
=================================================================
Total params: 22,156
Trainable params: 22,156
Non-trainable params: 0
_________________________________________________________________
```

```
Model: "gru_spectrogram"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
the_input (InputLayer)       [(None, None, 161)]       0
_________________________________________________________________
rnn (GRU)                    (None, None, 29)          16704
_________________________________________________________________
softmax (Activation)         (None, None, 29)          0
=================================================================
Total params: 16,704
Trainable params: 16,704
Non-trainable params: 0
_________________________________________________________________
```

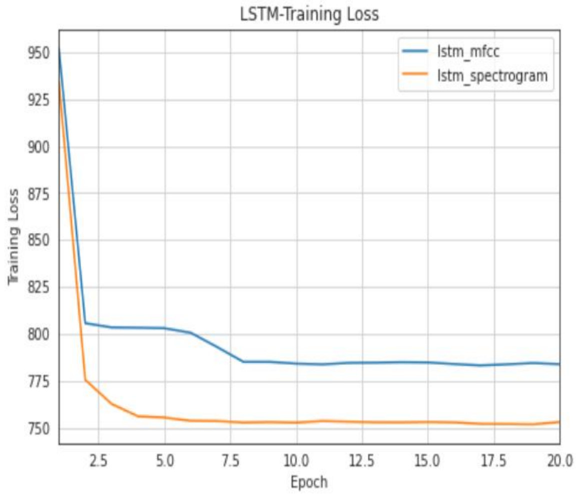
Models – Single Layer RNN – Result

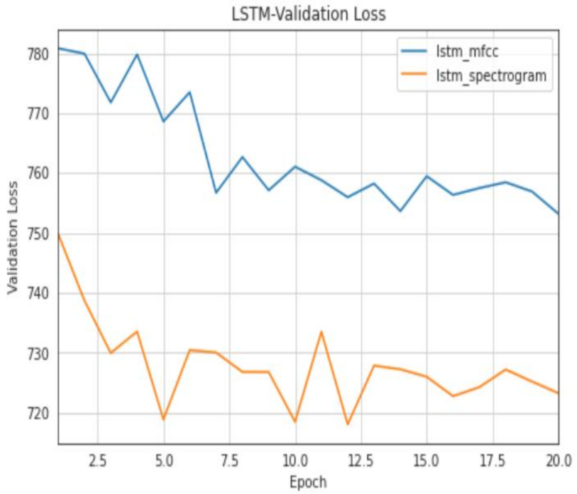
SimpleRNN



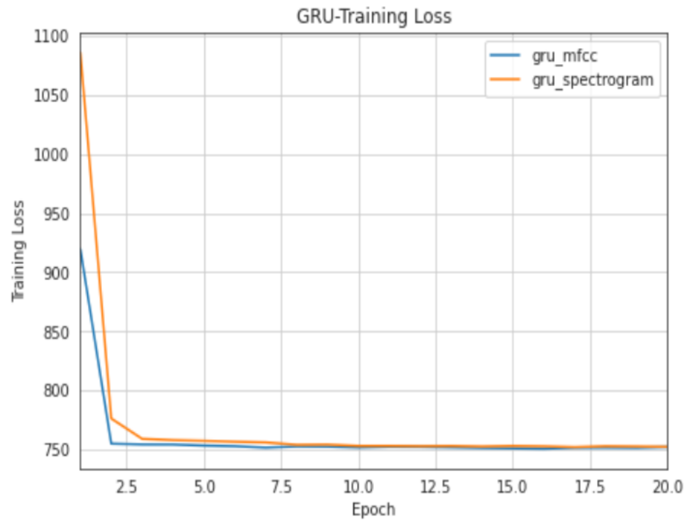


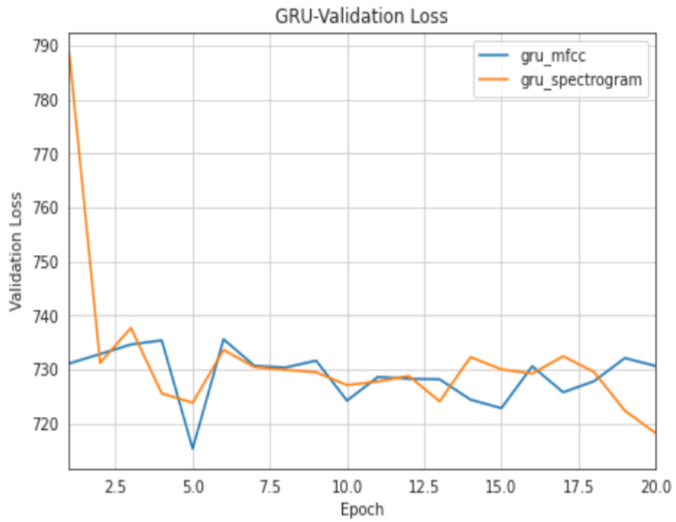
LSTM





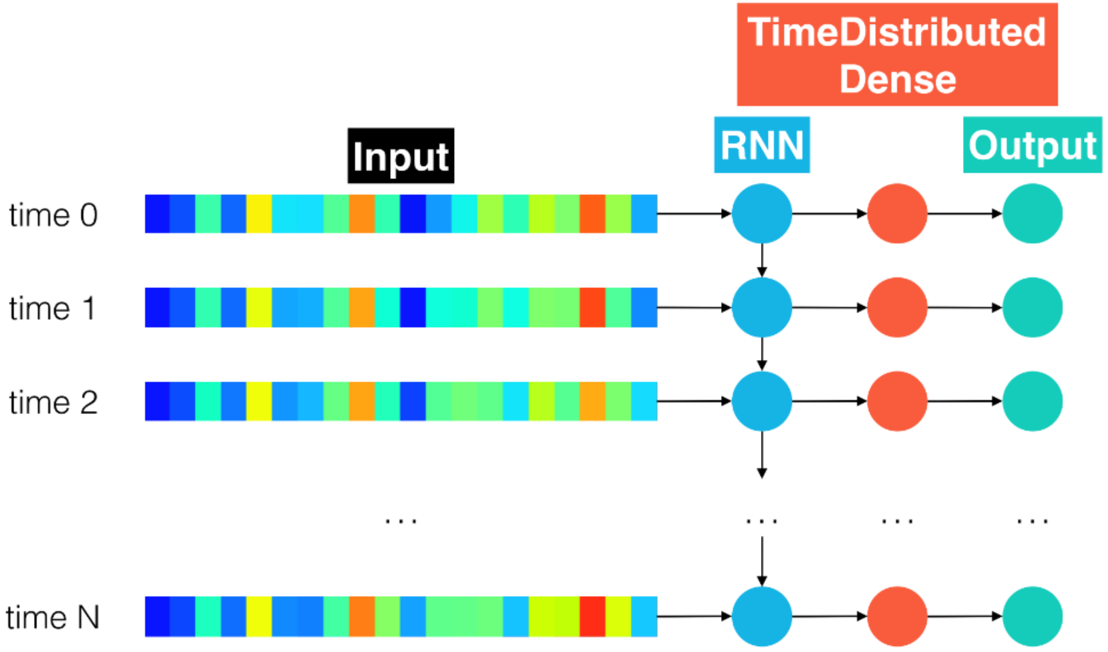
GRU

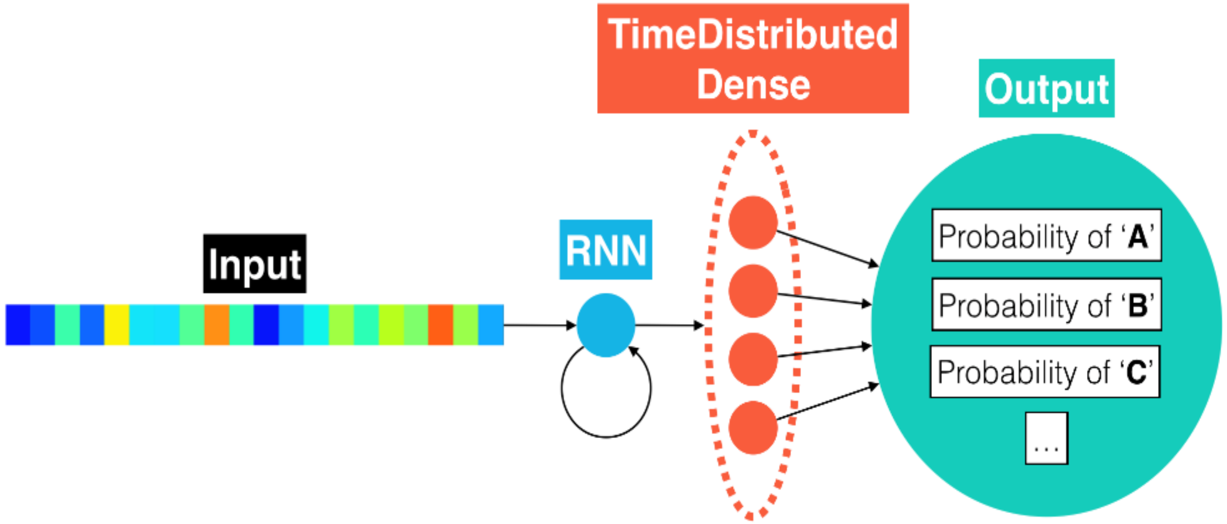




Model	Feature	Training Loss	Validation Loss
SimpleRNN	MFCC	754	721
	Spectrogram	752	735
LSTM	MFCC	783	753
	Spectrogram	752	723
GRU	MFCC	752	730
	Spectrogram	752	718

Models – RNN + TDD – Arch





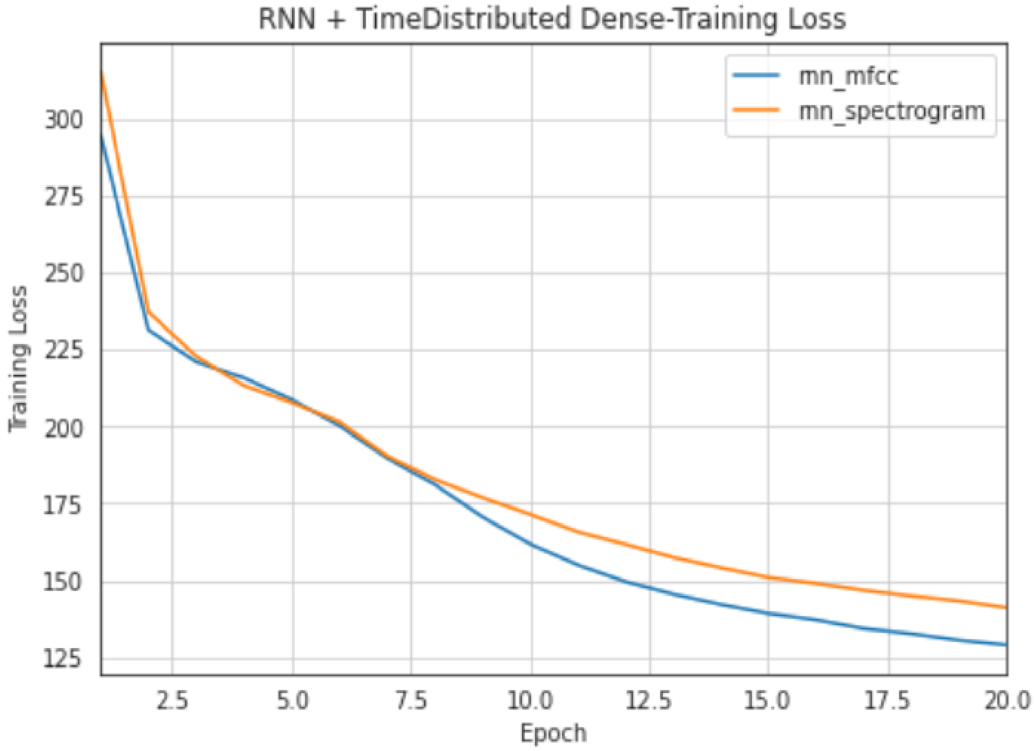
- TimeDistributed Dense wrapper and the BatchNormalization layer are applied to the GRU RNN layer
- Batch Normalization reduces training times
- Time Distributed layer helps find more complex patterns in the data set. Speeds up RNN, but is more memory intensive

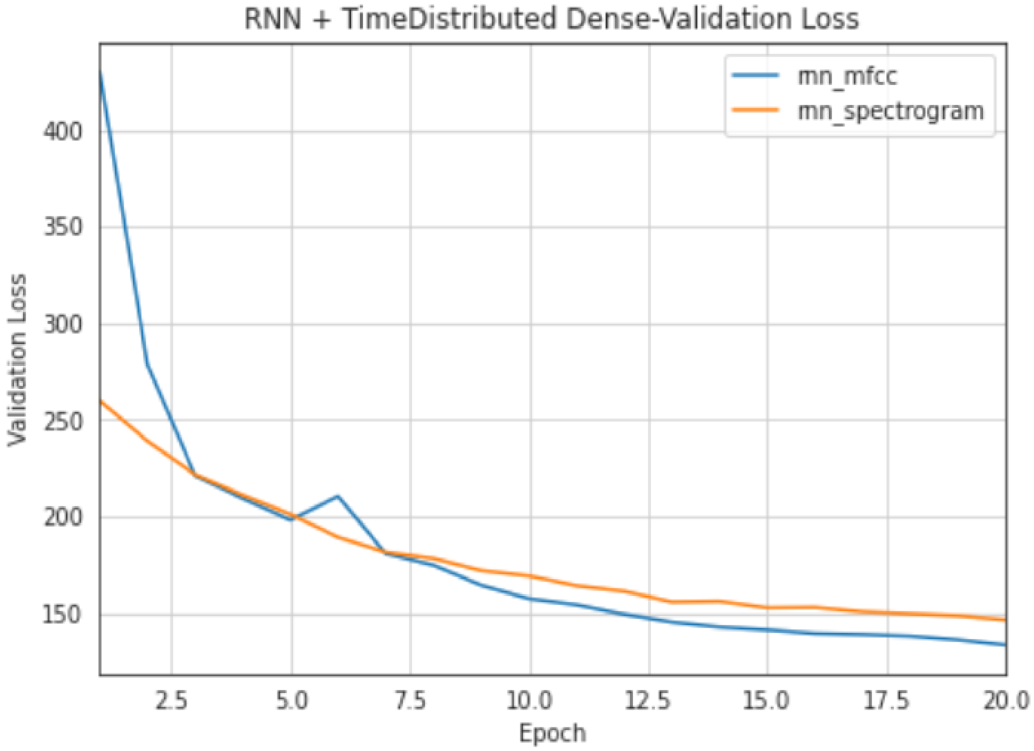
Models – RNN + TDD – Summary

```
Model: "rnn_mfcc"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
the_input (InputLayer)       [(None, None, 13)]        0
_________________________________________________________________
rnn (GRU)                    (None, None, 200)         129000
_________________________________________________________________
bn_rnn_1d (BatchNormalizatio (None, None, 200)         800
_________________________________________________________________
time_distributed_2 (TimeDist (None, None, 29)          5829
_________________________________________________________________
softmax (Activation)         (None, None, 29)          0
=================================================================
Total params: 135,629
Trainable params: 135,229
Non-trainable params: 400
_________________________________________________________________
```

```
Model: "rnn_spectrogram"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
the_input (InputLayer)       [(None, None, 161)]       0
_________________________________________________________________
rnn (GRU)                    (None, None, 200)         217800
_________________________________________________________________
bn_rnn_1d (BatchNormalizatio (None, None, 200)         800
_________________________________________________________________
time_distributed_3 (TimeDist (None, None, 29)          5829
_________________________________________________________________
softmax (Activation)         (None, None, 29)          0
=================================================================
Total params: 224,429
Trainable params: 224,029
Non-trainable params: 400
_________________________________________________________________
```

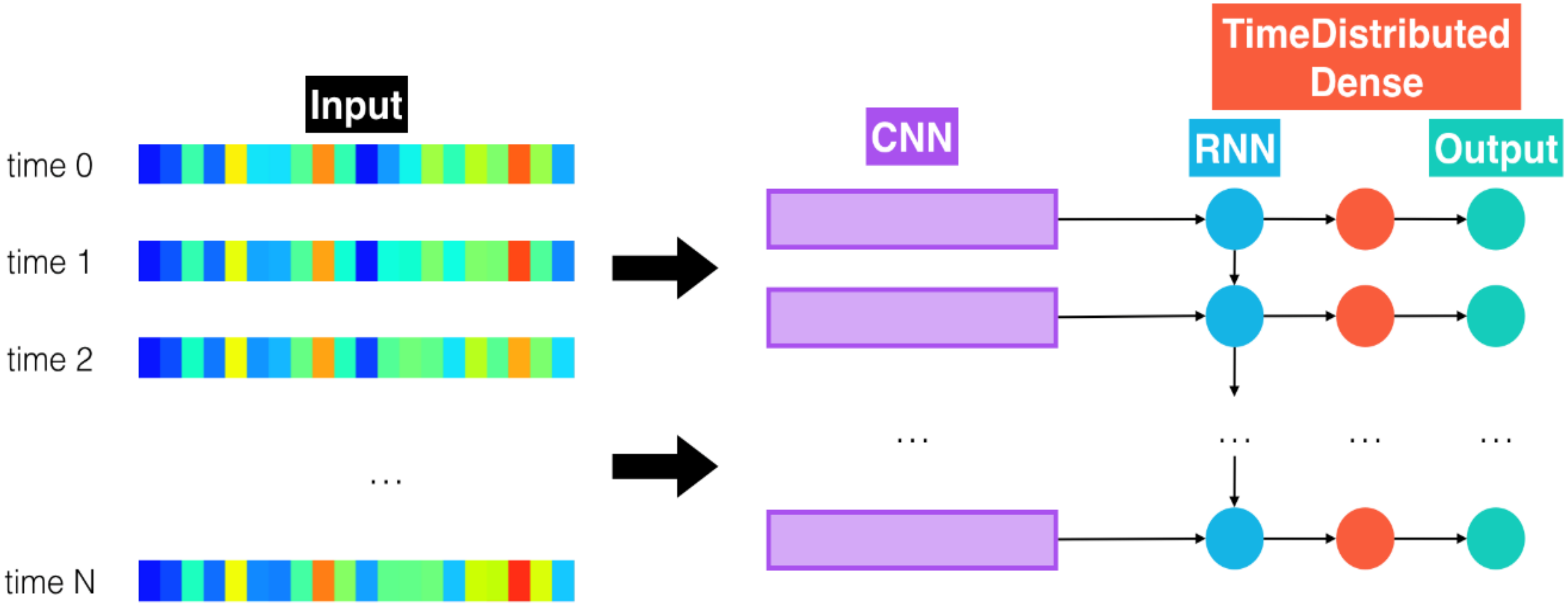
Models – RNN + TDD – Result





Feature	Training Loss	Validation Loss
MFCC	128	133
Spectrogram	141	146

Models – CNN + RNN + TDD – Arch



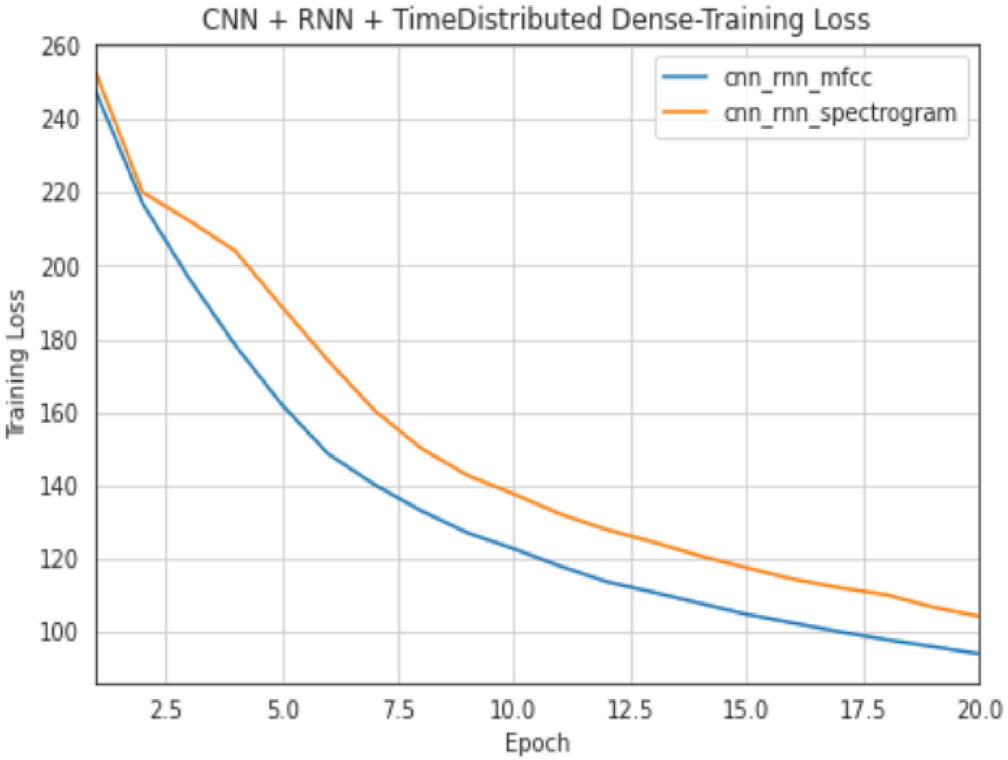
- Additional 1-D Convolutional Layer is added to increase the level of complexity

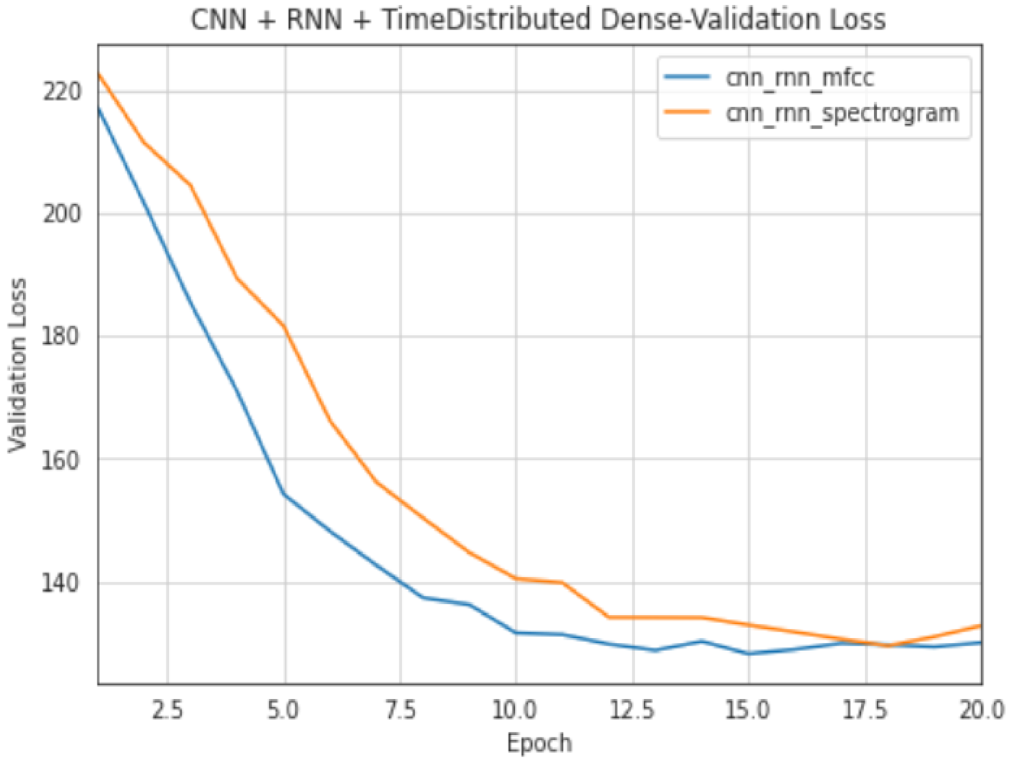
Models – CNN + RNN + TDD – Summary

```
Model: "cnn_rnn_mfcc"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
the_input (InputLayer)       [(None, None, 13)]        0         
_________________________________________________________________
conv1d (Conv1D)              (None, None, 200)         28800     
_________________________________________________________________
bn_conv_1d (BatchNormalizati (None, None, 200)         800       
_________________________________________________________________
rnn (GRU)                    (None, None, 200)         241200    
_________________________________________________________________
gru_rnn (BatchNormalization) (None, None, 200)         800       
_________________________________________________________________
time_distributed_4 (TimeDist (None, None, 29)          5829      
_________________________________________________________________
softmax (Activation)         (None, None, 29)          0         
=================================================================
Total params: 277,429
Trainable params: 276,629
Non-trainable params: 800
_________________________________________________________________
```

```
Model: "cnn_rnn_spectrogram"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
the_input (InputLayer)       [(None, None, 161)]       0         
_________________________________________________________________
conv1d (Conv1D)              (None, None, 200)         354400    
_________________________________________________________________
bn_conv_1d (BatchNormalizati (None, None, 200)         800       
_________________________________________________________________
rnn (GRU)                    (None, None, 200)         241200    
_________________________________________________________________
gru_rnn (BatchNormalization) (None, None, 200)         800       
_________________________________________________________________
time_distributed_5 (TimeDist (None, None, 29)          5829      
_________________________________________________________________
softmax (Activation)         (None, None, 29)          0         
=================================================================
Total params: 603,029
Trainable params: 602,229
Non-trainable params: 800
_________________________________________________________________
```

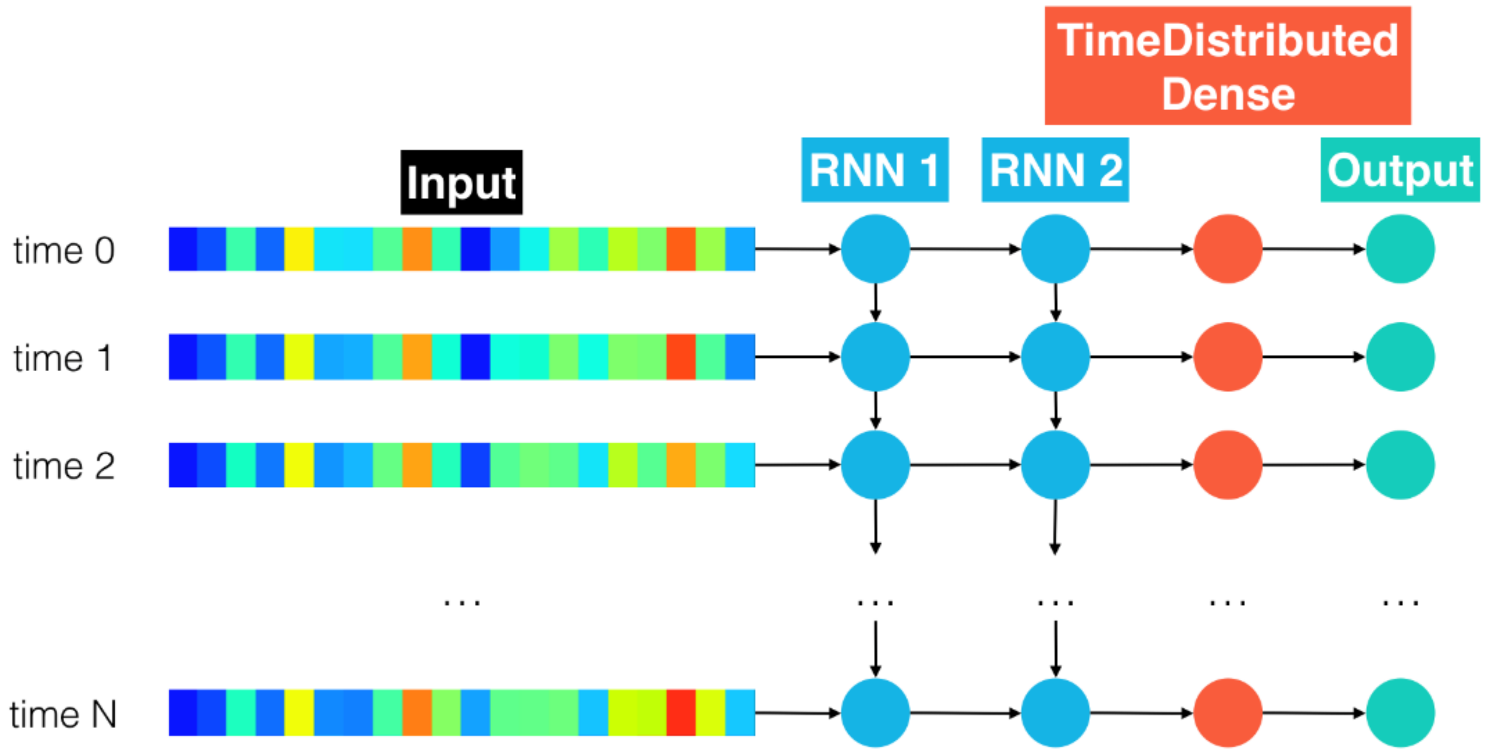
Models – CNN + RNN + TDD – Result





Feature	Training Loss	Validation Loss
MFCC	94	129
Spectrogram	104	132

Models – Deep RNN + TDD – Arch



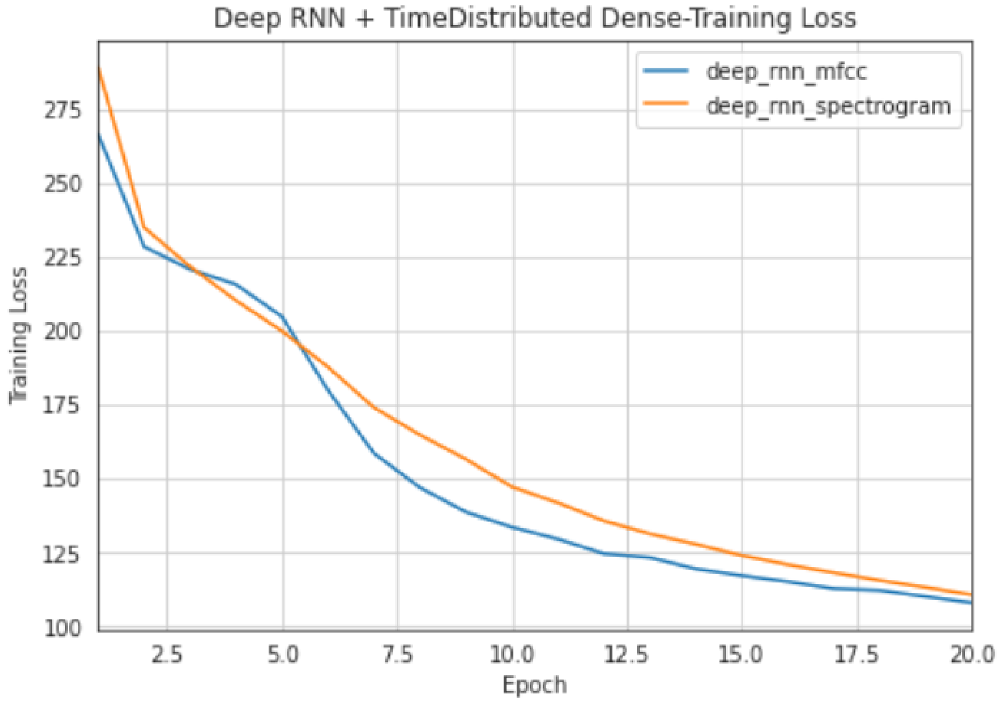
- A number of GRU to be added back to back which can process long sequences and interdependencies

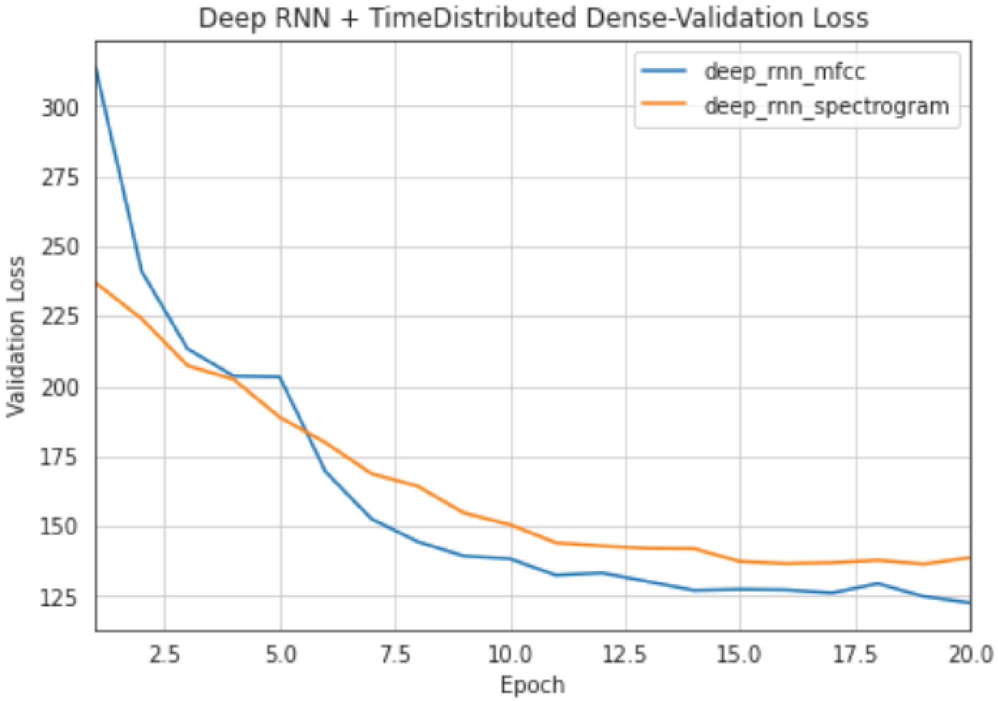
Models – Deep RNN + TDD – Summary

```
Model: "deep_rnn_mfcc"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
the_input (InputLayer)       [(None, None, 13)]        0
_________________________________________________________________
gru (GRU)                    (None, None, 200)         129000
_________________________________________________________________
bt_rnn_1 (BatchNormalization (None, None, 200)         800
_________________________________________________________________
gru_1 (GRU)                  (None, None, 200)         241200
_________________________________________________________________
bt_rnn_last_rnn (BatchNormal (None, None, 200)         800
_________________________________________________________________
time_distributed_6 (TimeDist (None, None, 29)          5829
_________________________________________________________________
softmax (Activation)         (None, None, 29)          0
=================================================================
Total params: 377,629
Trainable params: 376,829
Non-trainable params: 800
_________________________________________________________________
```

```
Model: "deep_rnn_spectrogram"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
the_input (InputLayer)       [(None, None, 161)]       0
_________________________________________________________________
gru_2 (GRU)                  (None, None, 200)         217800
_________________________________________________________________
bt_rnn_1 (BatchNormalization (None, None, 200)         800
_________________________________________________________________
gru_3 (GRU)                  (None, None, 200)         241200
_________________________________________________________________
bt_rnn_last_rnn (BatchNormal (None, None, 200)         800
_________________________________________________________________
time_distributed_7 (TimeDist (None, None, 29)          5829
_________________________________________________________________
softmax (Activation)         (None, None, 29)          0
=================================================================
Total params: 466,429
Trainable params: 465,629
Non-trainable params: 800
_________________________________________________________________
```

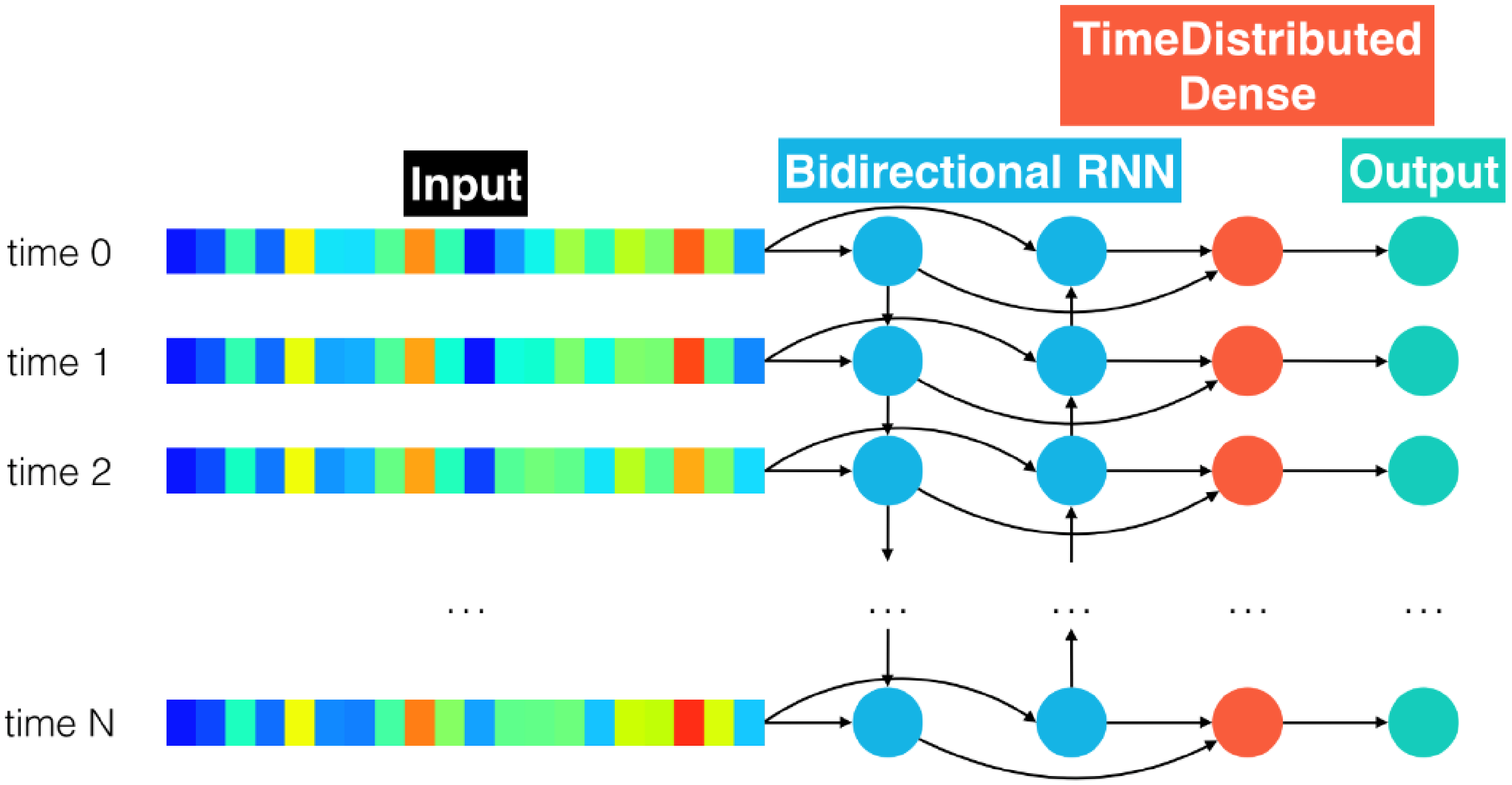
Models – Deep RNN + TDD – Result





Feature	Training Loss	Validation Loss
MFCC	108	122
Spectrogram	110	138

Models – Bidirectional RNN + TDD – Arch



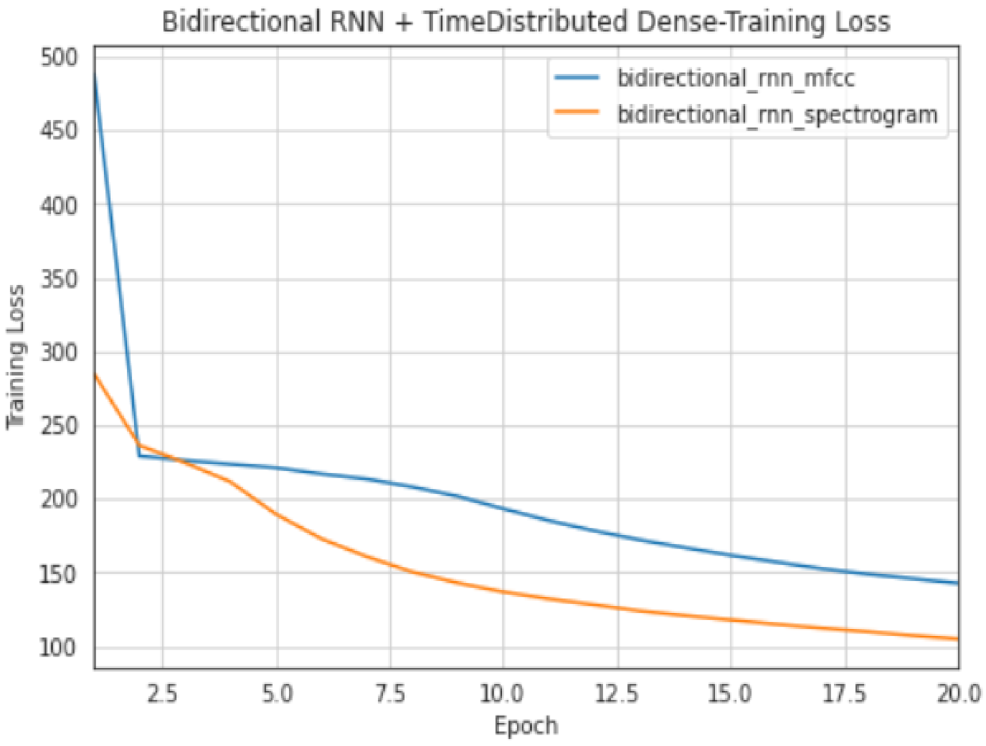
- Processes the data in both directions (forward and backward)
- Allows to assess future context.

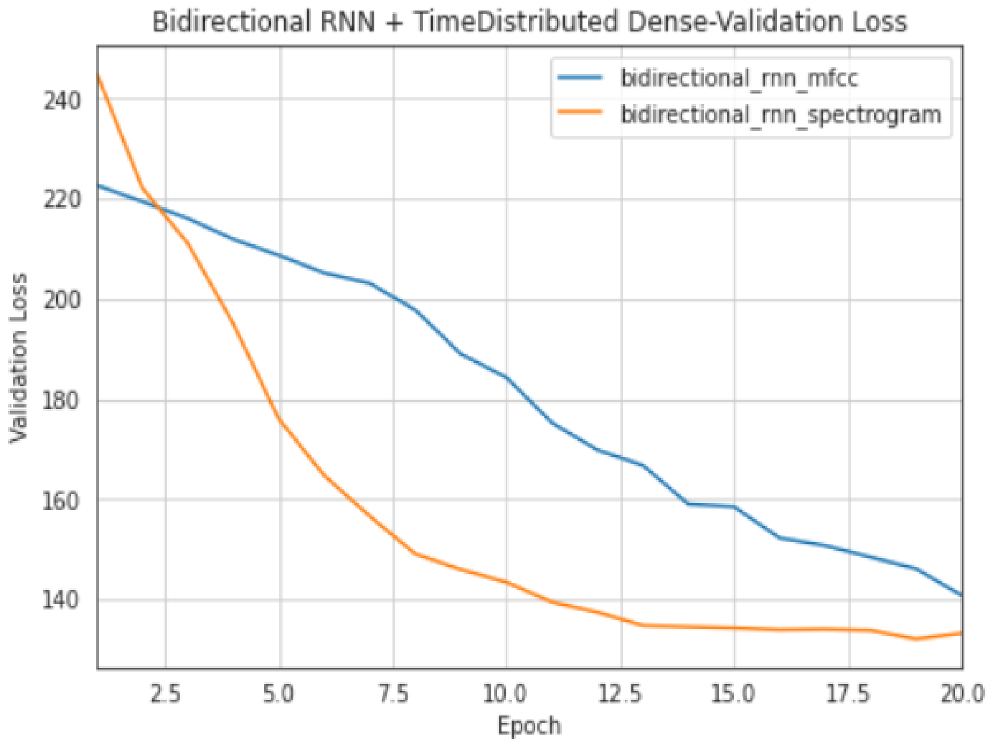
Models – Bidirectional RNN + TDD – Summary

```
Model: "bidirectional_rnn_mfcc"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
the_input (InputLayer)       [(None, None, 13)]        0
_________________________________________________________________
bidirectional (Bidirectional (None, None, 400)         258000
_________________________________________________________________
time_distributed_8 (TimeDist (None, None, 29)          11629
_________________________________________________________________
softmax (Activation)         (None, None, 29)          0
=================================================================
Total params: 269,629
Trainable params: 269,629
Non-trainable params: 0
_________________________________________________________________
```

```
Model: "bidirectional_rnn_spectrogram"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
the_input (InputLayer)       [(None, None, 161)]       0
_________________________________________________________________
bidirectional_1 (Bidirection (None, None, 400)         435600
_________________________________________________________________
time_distributed_9 (TimeDist (None, None, 29)          11629
_________________________________________________________________
softmax (Activation)         (None, None, 29)          0
=================================================================
Total params: 447,229
Trainable params: 447,229
Non-trainable params: 0
_________________________________________________________________
```

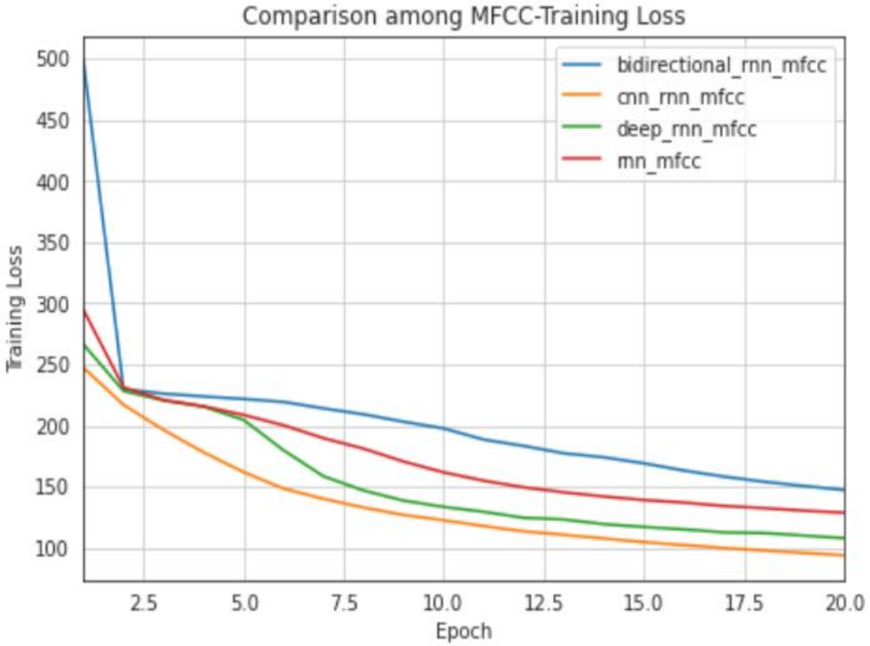
Models – Bidirectional RNN + TDD – Result

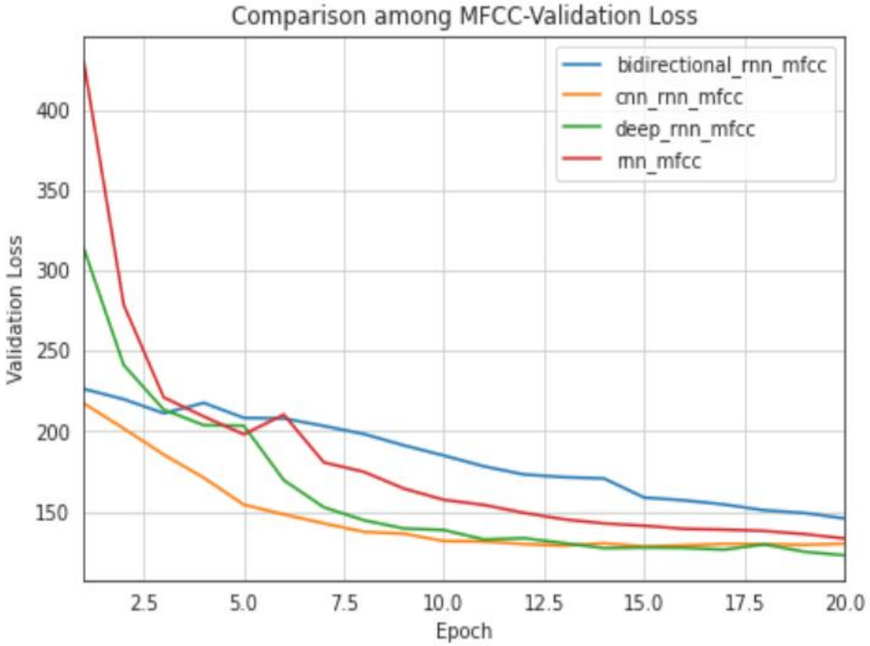


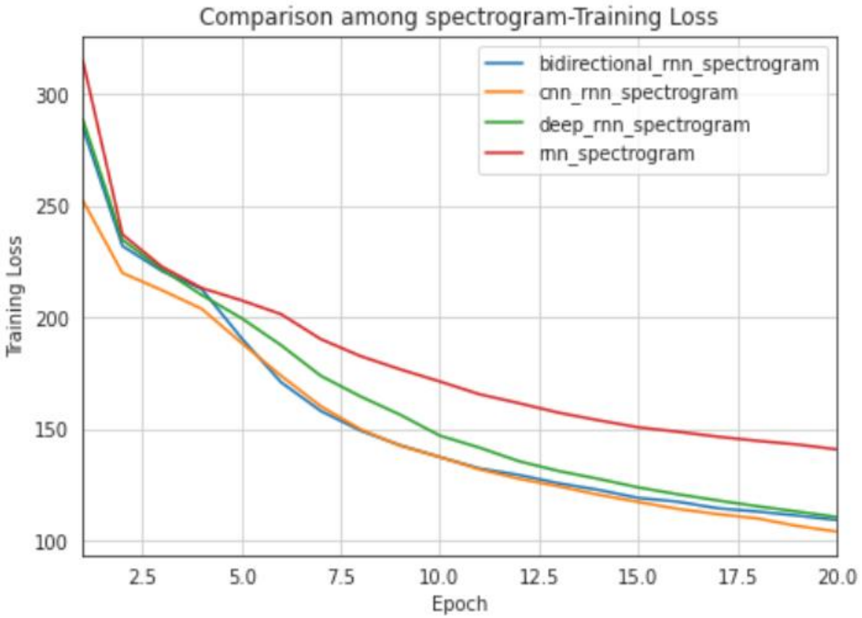


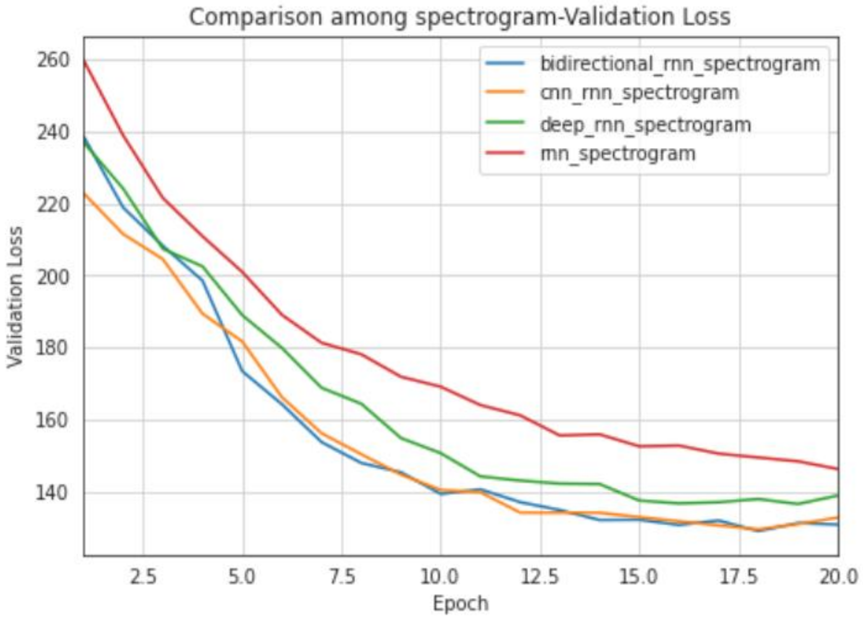
Feature	Training Loss	Validation Loss
MFCC	147	145
Spectrogram	109	130

Models – Comparison

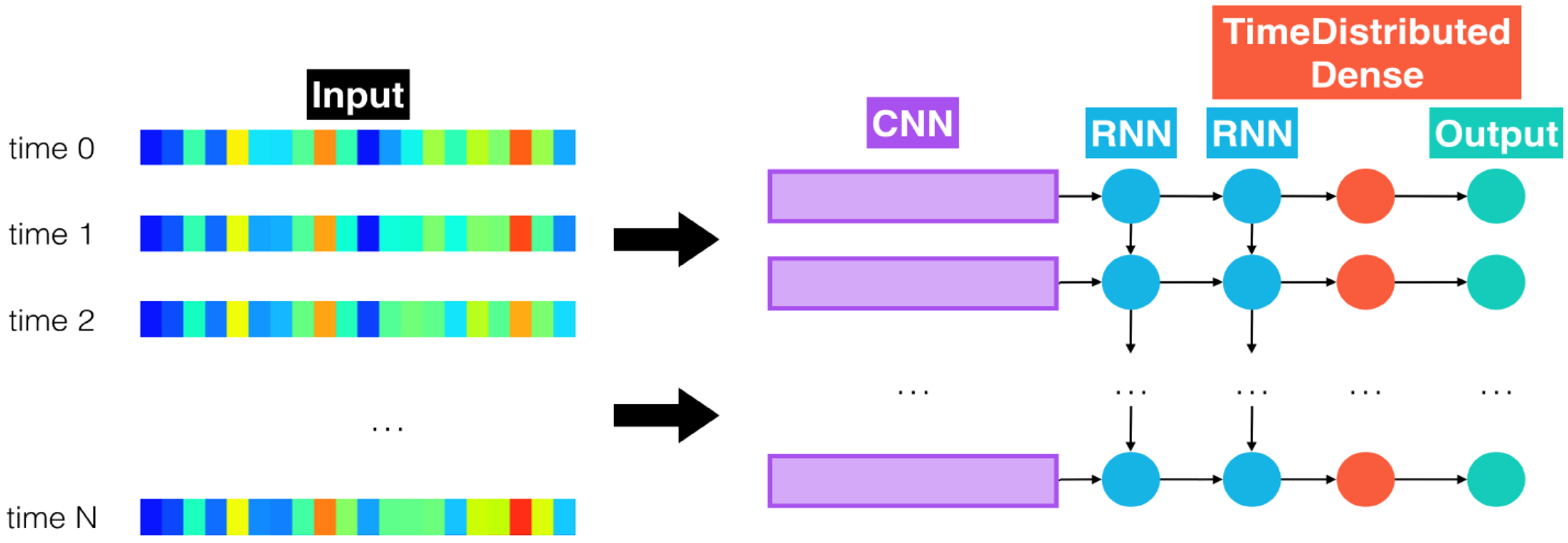








Models – Final – Arch



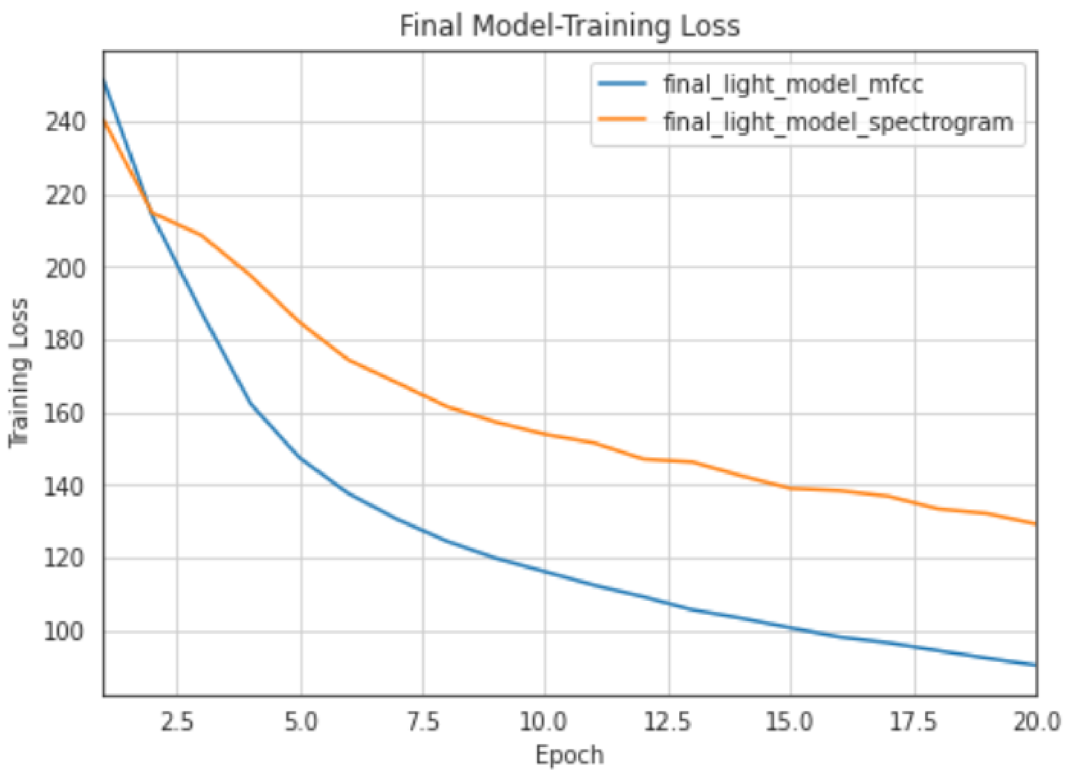
- Combination of CNN & Deep RNN with TimeDistributed Dense layer

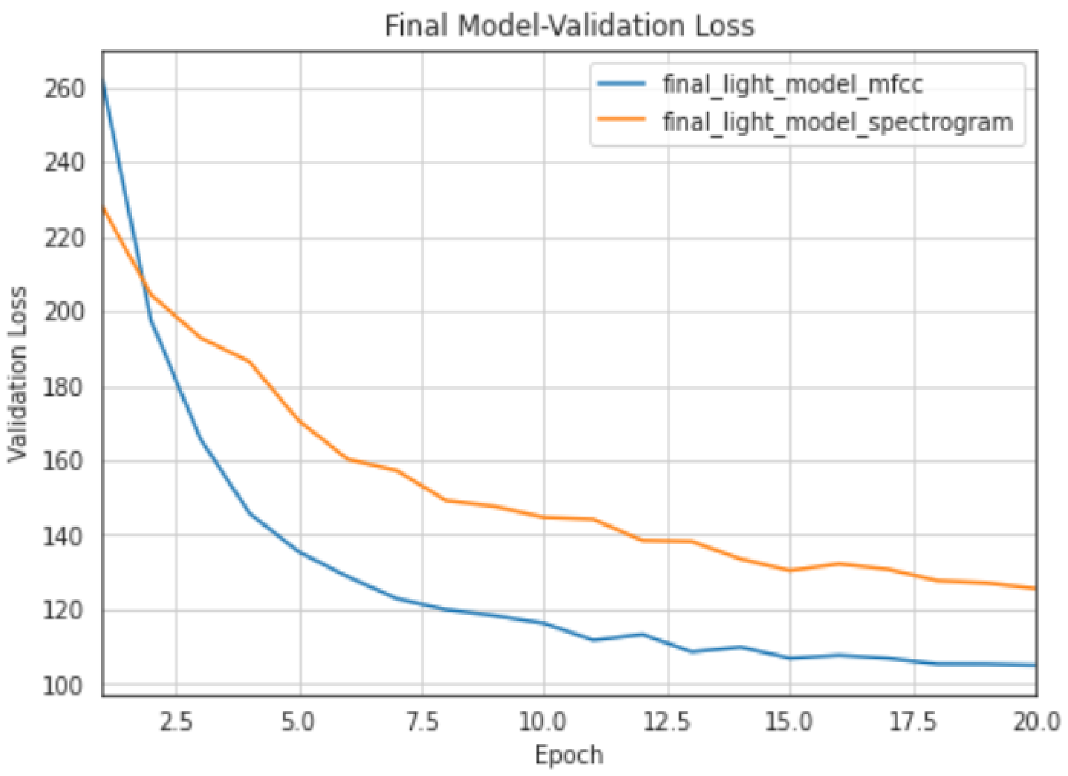
Models – Final – Summary

```
Model: "final_model_mfcc"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
the_input (InputLayer)       [(None, None, 13)]        0
_________________________________________________________________
layer_1_conv (Conv1D)        (None, None, 200)         28800
_________________________________________________________________
conv_batch_norm (BatchNormal (None, None, 200)         800
_________________________________________________________________
rnn_1 (GRU)                  (None, None, 250)         339000
_________________________________________________________________
bt_rnn_1 (BatchNormalization (None, None, 250)         1000
_________________________________________________________________
final_layer_of_rnn (GRU)     (None, None, 250)         376500
_________________________________________________________________
bt_rnn_final (BatchNormaliza (None, None, 250)         1000
_________________________________________________________________
time_distributed (TimeDistri (None, None, 29)          7279
_________________________________________________________________
softmax (Activation)         (None, None, 29)          0
=================================================================
Total params: 754,379
Trainable params: 752,979
Non-trainable params: 1,400
_________________________________________________________________
```

```
Model: "final_model_spectrogram"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
the_input (InputLayer)       [(None, None, 161)]       0
_________________________________________________________________
layer_1_conv (Conv1D)        (None, None, 200)         354400
_________________________________________________________________
conv_batch_norm (BatchNormal (None, None, 200)         800
_________________________________________________________________
rnn_1 (GRU)                  (None, None, 250)         339000
_________________________________________________________________
bt_rnn_1 (BatchNormalization (None, None, 250)         1000
_________________________________________________________________
final_layer_of_rnn (GRU)     (None, None, 250)         376500
_________________________________________________________________
bt_rnn_final (BatchNormaliza (None, None, 250)         1000
_________________________________________________________________
time_distributed_1 (TimeDist (None, None, 29)          7279
_________________________________________________________________
softmax (Activation)         (None, None, 29)          0
=================================================================
Total params: 1,079,979
Trainable params: 1,078,579
Non-trainable params: 1,400
_________________________________________________________________
```

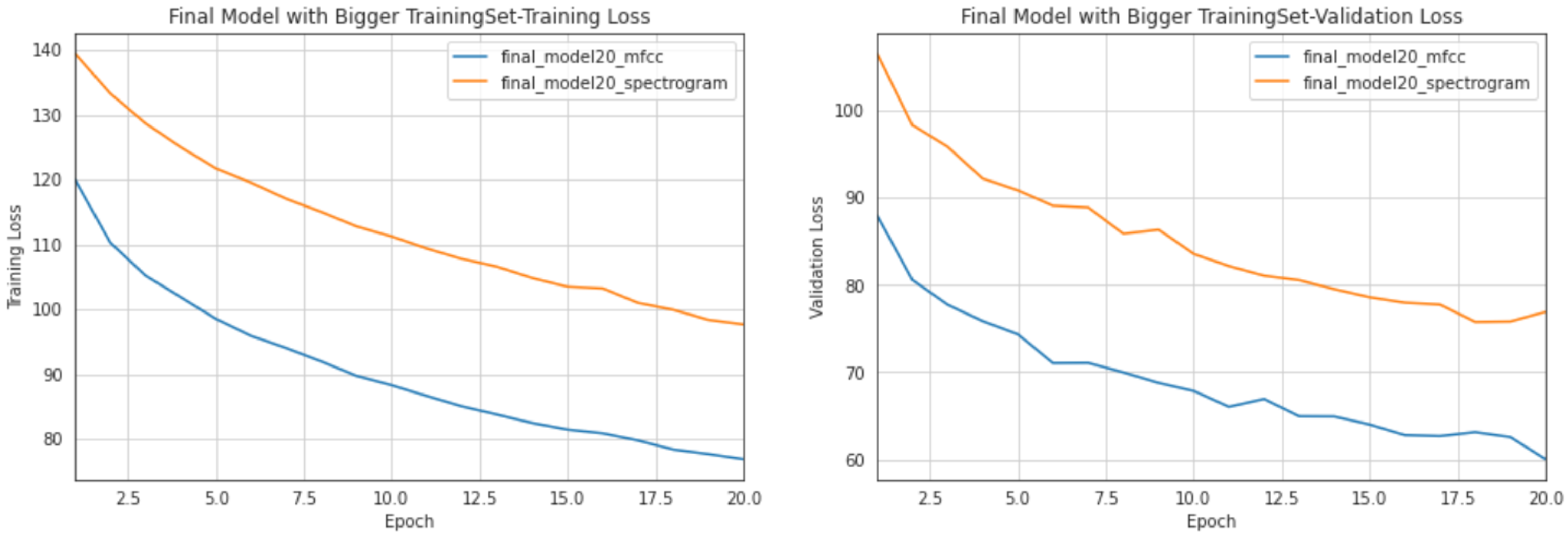
Models – Final – Result





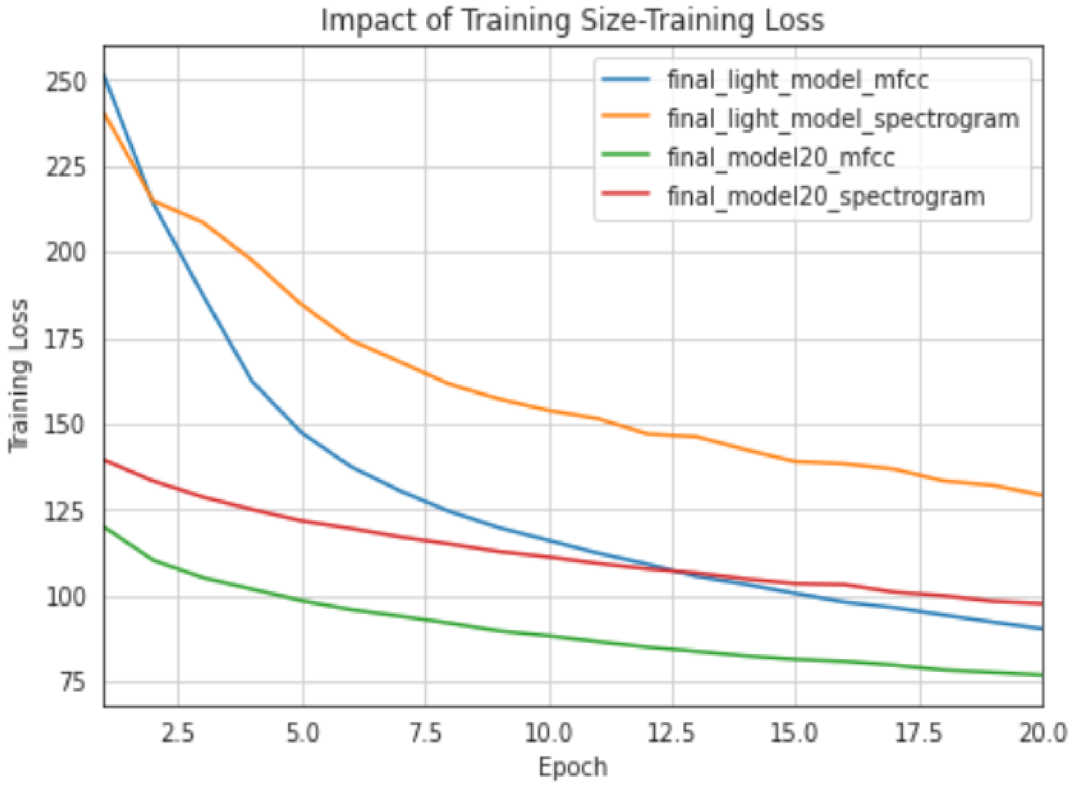
Feature	Training Loss	Validation Loss
MFCC	90	105
Spectrogram	129	125

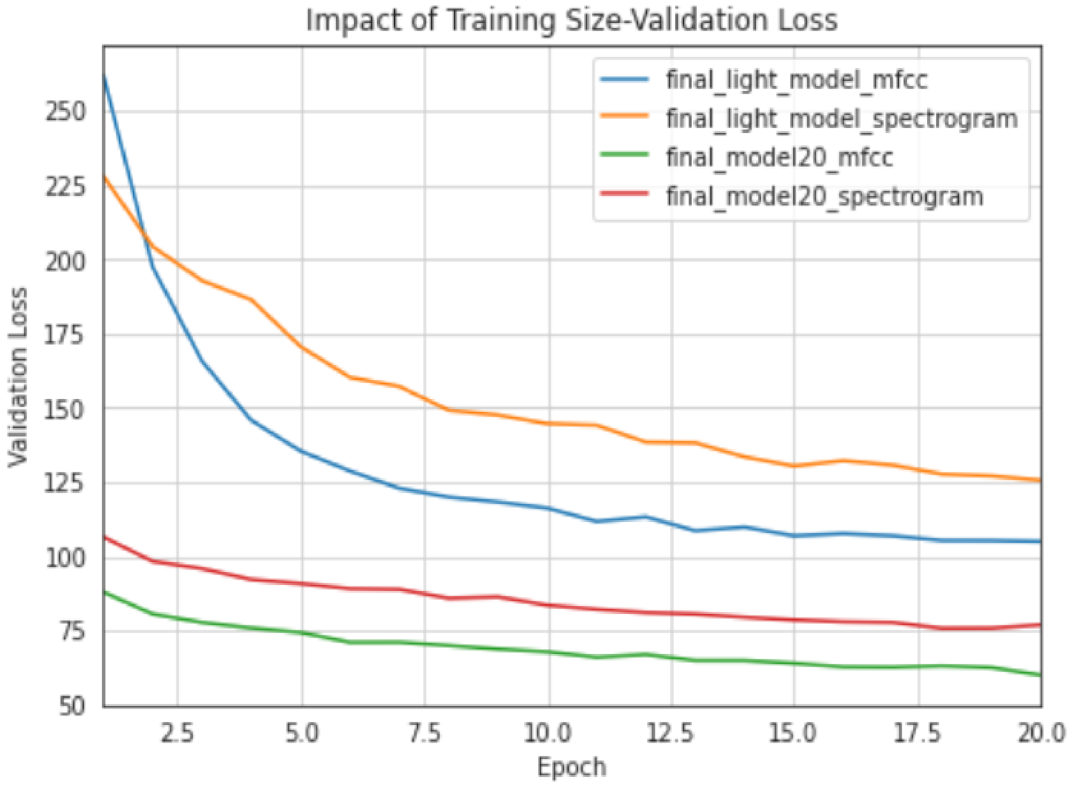
Models – Final - with train-360-clean



Feature	Training Loss	Validation Loss
MFCC	77	60
Spectrogram	98	77

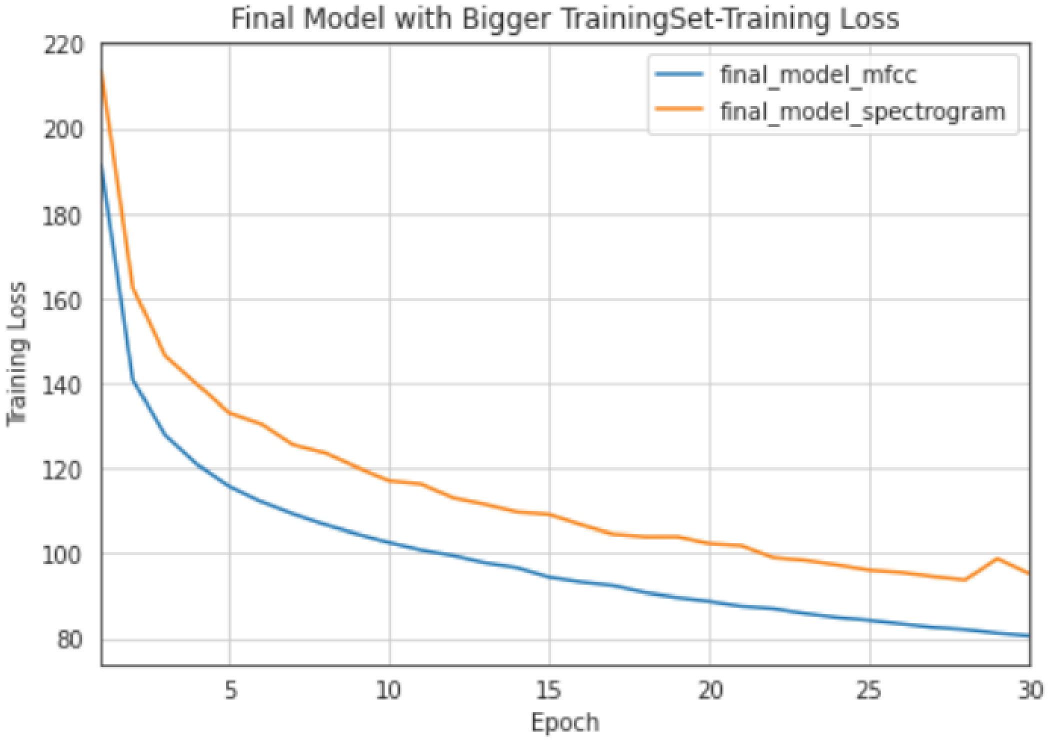
Models – Final- Impact of Training Size

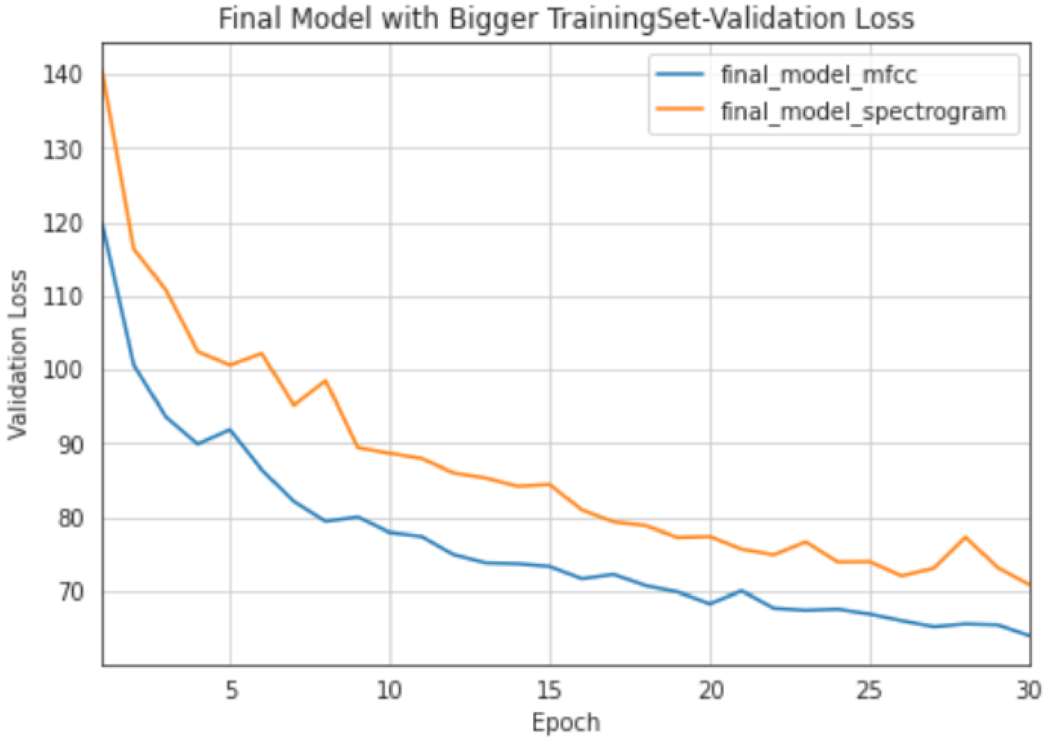




- Loss reduced with increase of training size

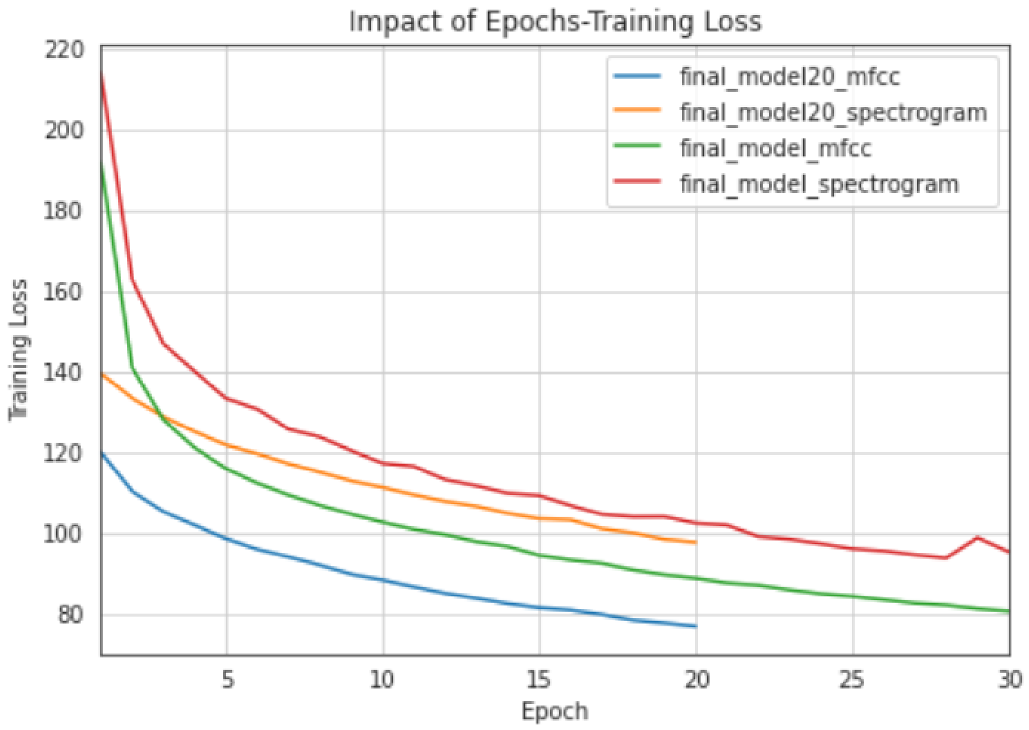
Models – Final- with train-360 and 30 epochs

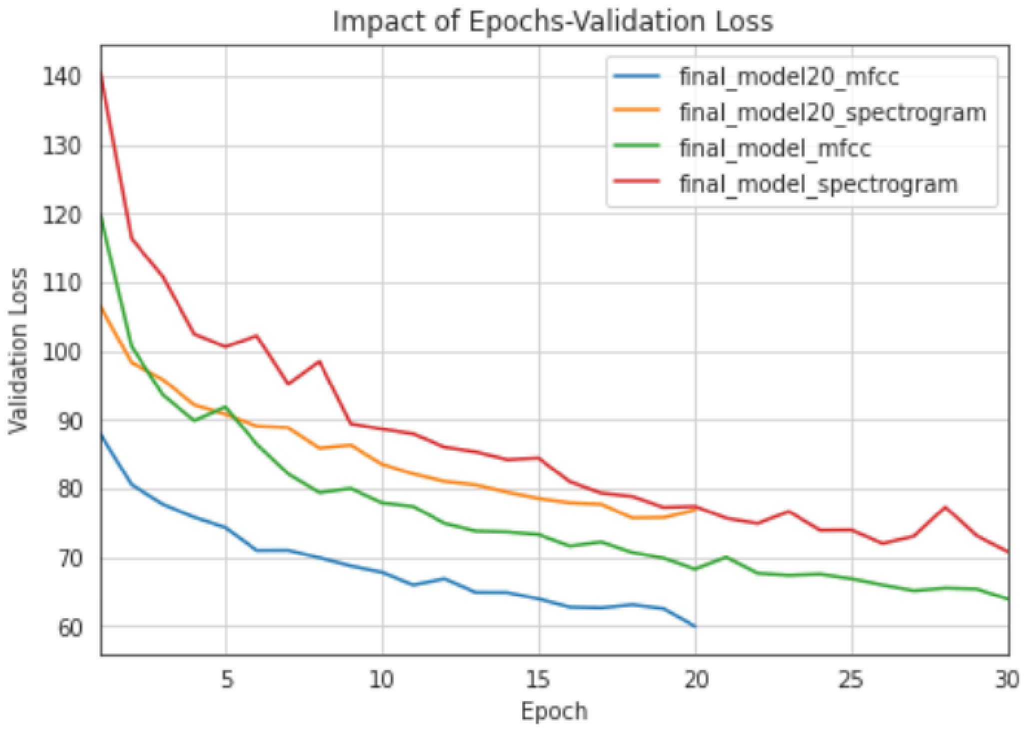




Feature	Training Loss	Validation Loss
MFCC	80	64
Spectrogram	95	71

Models – Final- Impact of Epochs





- Slight decrease in loss when number of epochs increased (yet to reach plateau)

Predictions using Final Model

```
from model_use import ModelPredictor
predictor = ModelPredictor(input_to_softmax=final_model_mfcc,model_path='results/final_model_mfcc.h5')
from IPython.display import Audio

# display the true and predicted transcriptions and show the audio file
predictor.get_predictions_index(index=0,
                partition='train',
                spectrogram = False)
print('-'*80)
Audio(predictor.audio_path)
```

```
True transcription:

and now it had come to pass that his sole remaining ally mister samuel bozzle the ex policeman was becomin
g weary of his service
--------------------------------------------------------------------------------
Predicted transcription:

ands now had com to pas atd his so remaning oli mister san no bossl the axpilisemen was becomeing wear yef
his servace
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
```

```
from model_use import ModelPredictor
predictor = ModelPredictor(input_to_softmax=final_model_mfcc,model_path='results/final_model_mfcc.h5')
from IPython.display import Audio

# display the true and predicted transcriptions and show the audio file
predictor.get_predictions_index(index=0,
                partition='validation',
                spectrogram = False)
print('-'*80)
Audio(predictor.audio_path)
```

```
miss lake declined the carriage to night
--------------------------------------------------------------------------------
Predicted transcription:

misrak to cines te carged to mat
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
```

Real-Time Prediction Engine

```
from model_use import ModelPredictor
from IPython.display import Audio
from voice_rec import voice_record

predictor = ModelPredictor(input_to_softmax=final_model_mfcc,model_path='results/final_model_mfcc.h5')
voice_record(path='recordings/demo_mfcc.wav')
# display the true and predicted transcriptions and show the audio file
predictor.get_predictions_recorded(spectrogram = False,recordingpath='recordings/demo_mfcc.wav')
print('-'*80)
Audio(predictor.audio_path)
```

```
please speak into the microphone
done - result written to recordings/demo_spectro.wav
--------------------------------------------------------------------------------
Predicted transcription:

lis go
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
```

"Let's go"

```
from model_use import ModelPredictor
from IPython.display import Audio
from voice_rec import voice_record

predictor = ModelPredictor(input_to_softmax=final_model_spectrogram,model_path='results/final_model_spectr
ogram.h5')
# record your voice
voice_record(path='recordings/demo_spectro.wav')
# display the true and predicted transcriptions and show the audio file
predictor.get_predictions_recorded(spectrogram = True,recordingpath='recordings/demo_spectro.wav')
print('-'*80)
Audio(predictor.audio_path)
```

```
please speak into the microphone
done - result written to recordings/demo_spectro.wav
--------------------------------------------------------------------------------
Predicted transcription:

a a you
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
```

"How are you"

Conclusions

MFCC as a feature outperforms Spectrogram in accuracy of the models.

CNN layer ahead of GRU RNN improved the accuracy significantly.

Increasing the depth of RNN layer also increased accuracy.

Combining the best performing models (CNN+RNN+TDD and Deep RNN + TDD) further reduced the loss values.

Model quality can be further improved by training with bigger training set and increasing the number of epochs.

Adam optimizer provide a better stability for training deep learning network due to its self-stabilizing nature.

Future Work

Assessment of additional layers

- Additional combinations of Keras layers can be tried to further expand the possibility of a better model. An example would be to use deep bidirectional RNNs.

Beam Size Impact

- Controlling the beam size for CTC batch can be analyzed to see the impact of beam size on accuracy.

Noise reduction

- Noise reduction code can be implemented in the voice recording module to generate better real-time recognitions.

Lexicon Language Model

- Lexicon language model provided in the Librispeech can be used as the Labels for training. This will get a better recognition rate for words in dictionary.

References & Improvements

Base code : https://github.com/udacity/AIND-VUI-Capstone

Data set : http://www.openslr.org/12/

Features added to base code:

- Added Various Model Codes
- Voice Recording Module
- Tensor board integration
- Upgraded to latest versions of
  - TensorFlow
  - Python Packages
  - CUDA
- Visualization modules

GitHub Link: https://github.com/lithathampan/speech_recognition_nn_comparison.git

Thank You