CS 419M

VOICE CONVERSION

Final Report

Team Members

Arpan Banerjee	150070011
Nihal Singh	150040015
Srivatsan Sridhar	150070005

Abstract

The Voice Conversion task involves converting speech from one speaker's (source) voice to another speaker's (target) voice. Machine learning methods can be made to perform better than plain signal processing techniques as they can take into account multiple features of speech which cannot be characterized easily by signal processing techniques. In this project, we have explored the use of Recurrent Neural Networks (RNNs) for Voice Conversion. We have explored multiple variations of RNNs using LSTMs and GRUs and observed the effects of changing various parameters of the models. Our approach uses two independently trained neural networks - one which converts source speech to phonemes and another which converts phonemes to target speech. We will present the results achieved by both the networks for these different parameters.

Table of Contents

Introduction

Problem Description

This project aims to convert one person's (source) voice into another person's (target) voice. The two main questions about a voice conversion system are: 1) "How natural does the converted voice sound?" and 2) "How similar does the converted voice sound to the target voice and the source voice?". Earlier, signal processing was used to tackle this problem. However, with signal processing, only the pitch and frequencies of the voice are modulated. Applying machine learning techniques to this problem allows us to factor in other characteristics of speech such as stress on certain syllables and the timbre of the voice.

Applications of Voice Conversion

Voice Conversion finds its use in automatic dubbing for movies where one can change the speech into the desired actor's voice. Voice conversion is also useful in automatic speech translation. In the setting of an international conference, for example, the speech of a dignitary can be translated from a foreign language by a translator and then converted back to the original speaker's voice. It can also be used to impersonate somebody's voice, and can be used against another emerging technique that is voice encryption.

Model Architecture

We have used a sequence to sequence approach using Recurrent Neural Networks. The architecture is divided into two stages. The first stage (Net1) comprises of converting MFCCs (Mel Frequency Cepstral Coefficients) extracted from the source waveform to phonemes. These are fed into the next neural network (Net2) which converts phonemes to the target waveform.

We have tried different architectures for both the networks, including variations of LSTMs and GRUs. We have explored the effects of changes in the models such as varying the number of hidden layers, dropout rate and creating a pyramidal network structure. We have trained both the networks individually for these different cases and observed their effect.

Datasets

We have made use of the TIMIT dataset which has frame level phoneme transcriptions for utterances by 630 speakers, for training the first neural network. In addition, we've used the CMU Arctic dataset for training our second neural network. The Arctic dataset consists of 1150 utterances from a single male and female speaker (target).

Evaluation

The common method to evaluate a speech recognition model that converts waveform to phonemes consists of calculating the Phoneme Error Rate (PER). PER can be understood as the edit distance between two sequences, i.e. the minimal number of insertions, deletions and substitutions required to convert one sequence to another.

However, we are using Net1 to get phoneme labels per frame which are of importance while converting to target waveform. PER is calculated for the phoneme sequence (without repeats). The metric of accuracy we've used is a straightforward 1/0 approach i.e. whether the predicted phoneme at a frame matches the actual label or not. It is important to note here that the output for Net1 is a sequence of the length of the number of frames in the audio i.e. it consists of repeated phonemes, instead of the actual phoneme sequence over which PER can be calculated. For this reason, we've not been able to benchmark our results against state of the art.

For speech synthesis tasks, a metric known as the Mean Opinion Score (MOS) exists. Mean Opinion Score is a subjective measure of voice quality, and is a way to assess human users' opinion of the speech. It is a rating from 0 to 5 for sound quality averaged over a number of users. (1: Completely Unnatural, 2: Mostly Unnatural, 3: Equally Natural and Unnatural, 4: Mostly Natural, 5: Completely Natural)

We would be evaluating Net2 by just listening to the output audio. We will not be using an MOS score due to lack of an able panel of listeners who will give the scores.

State of the art

For the approaches, we've undertaken, i.e. Net1 which converts Waveform to Phonemes has a state of the art Phoneme Error Rate of 16.5% on TIMIT using Hierarchical maxout CNN + Dropout. [Ref: Toth et al.]

Net 2, best resembles DeepMind's WaveNet, which is a generative model to synthesize raw audio. The MOS score for WaveNet on US English is 4.55 and with Mandarin 4.21, making it very close to human level performance. [Ref: DeepMind's blog, paper]

Lately, state of the art for end to end voice conversion models use Generative Adversarial Networks (GANs). The Voice Conversion Challenge 2018 results show the best MOS of just above 4, for various tasks.

Existing Techniques

Tacotron (Wang et al.) is a text to speech end to end system achieving an MOS of 3.82. It introduces CBHGs (1-D Convolution Banks, Highway network and GRU) as an effective sequence to sequence model.

Attention mechanism is also used which calculates different weights for the input sequence, thus allowing the decoder to figure out which of the elements of the input sequence are most relevant.

Methodology Used

Background

Voice Conversion may be 1) *end-to-end voice conversion* or 2) a *combination of speech recognition and speech synthesis*.

End-to-end systems use a single system to convert the source voice to target voice. One method we studied uses multiple iterations of sequence-to-sequence models, using GRUs and phased LSTMs with attention mechanism. Some methods use a mid-level phonetic representation at the intermediate stage. One such method is to use a deep auto-encoder to obtain compact representations of the spectra of speakers, a neural network to convert them to another speaker, and an auto-decoder to convert it back to speech [S. H. Mohammadi and A. Kain]. End-to-end voice conversion, in general, requires parallel training data (same sentences spoken in source and target voices). It also requires some more techniques such as Dynamic Time Warping in order to align the similar frames of the source and target.

We have used the second type of method, where one network (Net1) converts the speech into phonemes (speech recognition) and another network (Net2) converts the phonemes into the target voice (speech synthesis). This method does not require parallel training data, so it does not impose such restrictions on the dataset. This method is also much simpler and easier to implement. Training becomes easy as the two networks can be trained and evaluated independently.

Net1 - Speech to Phonemes

Dataset

Net1 makes use of the wav files from the TIMIT dataset as input. These consist of utterances of 630 speakers, of phonetically rich sentences. There are a total of 4620 files constituting the Training set and 1680 files constituting the Test set. The wav files are sampled at 16000 Hz and are of varying lengths. We also have the corresponding phonemes along with their start and end times in the audio. These are converted to per frame phonemes where all the frames between the start and end time of a phoneme are labelled the same. A one hot representation of these phonemes forms the output of the network.

MFCCs

The first step is to compute the MFCCs of the waveform. MFCCs or Mel Frequency Cepstral Coefficients can be viewed as a representation of speech from which phoneme extraction is easier. MFCCs are popularly used in a variety of speech processing tasks because they strongly relate to human perception. The frame size used is 25ms and the hop size is 5ms. The following figure illustrates the calculation of MFCCs.

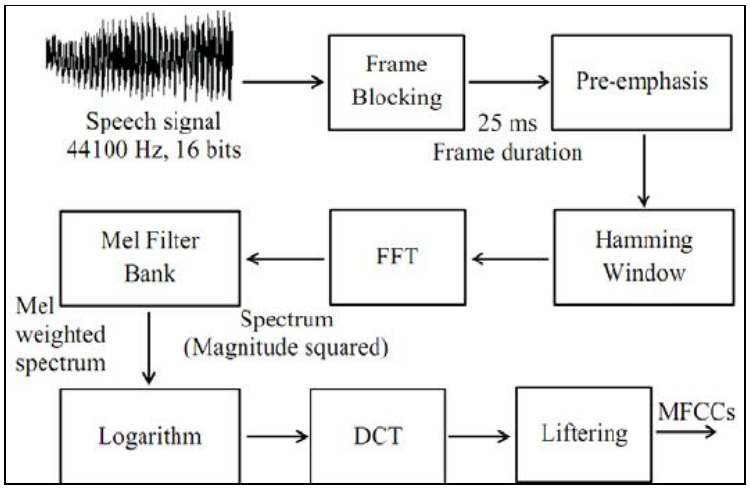

Image source: Detecting Patients with Parkinson's disease using MFCCs...

Preparing Input Data

The dataset consisted of files of varying lengths. For every frame, we had a total of 40 MFCC values. To feed the input we create an array of dimensions (BATCH_SIZE x TIME_FRAMES x 40), where BATCH_SIZE is the number of samples passed at once per batch, TIME_FRAMES is the number of frames per file and 40 is the number of MFCC features.

One issue we faced is that the wav files used as input are of varying length. Our first attempt was to pad the shorter wav files with zeros at the end. This had two problems : the input matrix now contained too many zeros, and we ran out of GPU memory. Thus we took an approach of selecting a number of random slices of 2 seconds length from the wav files. This forms our batch for training the network. The samples are resliced (to get different 2-second samples from the wav files) once every few epochs. The MFCCs are calculated for these samples, and they are normalized by subtracting the mean and dividing by the standard deviation of the training samples. This forms the input of the network.

Model Structure

Each hidden layer of Net1 consists of two Long Short Term Memory (LSTM) cells, one for the forward direction and one for the backward direction (bidirectional RNN). The figure below depicts this model. The number of layers and the number of units in each layer are parameters that we have varied across different tests. The output of the last layer is a probability distribution over the phoneme classes (known as PPGs - Phonetic PosteriorGrams). We then take an argmax to determine the most likely phoneme for each time frame.

The changes that we further made to this structure are:

- Using bidirectional networks instead of unidirectional
- Changing LSTM cells to GRU cells
- Changing the number of hidden layers and number of units in each layer
- Adding dropout to the network

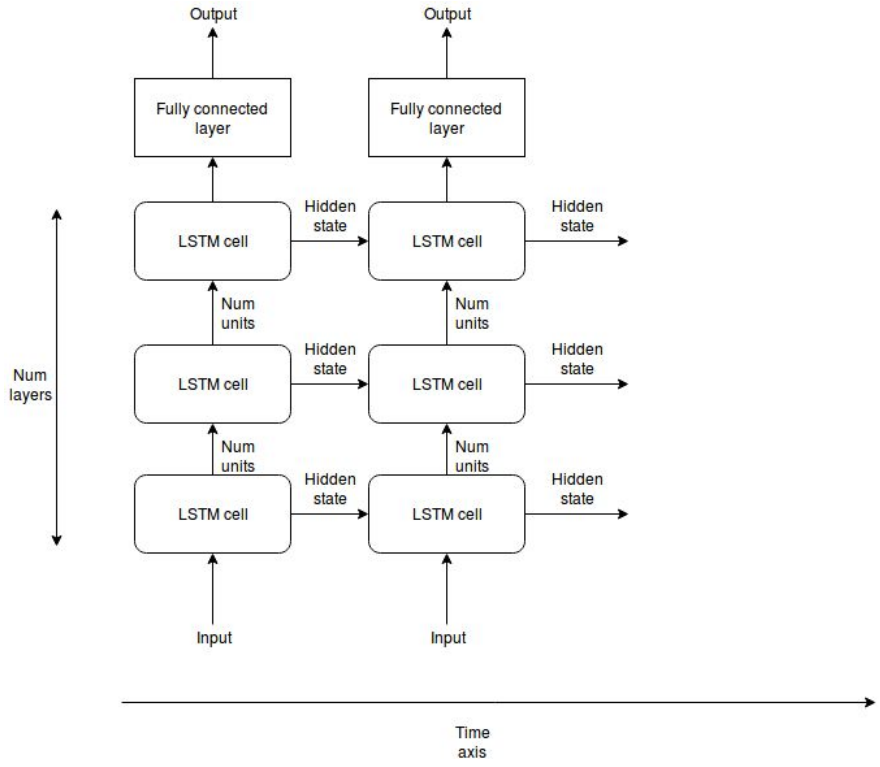

Figure: Net1 model

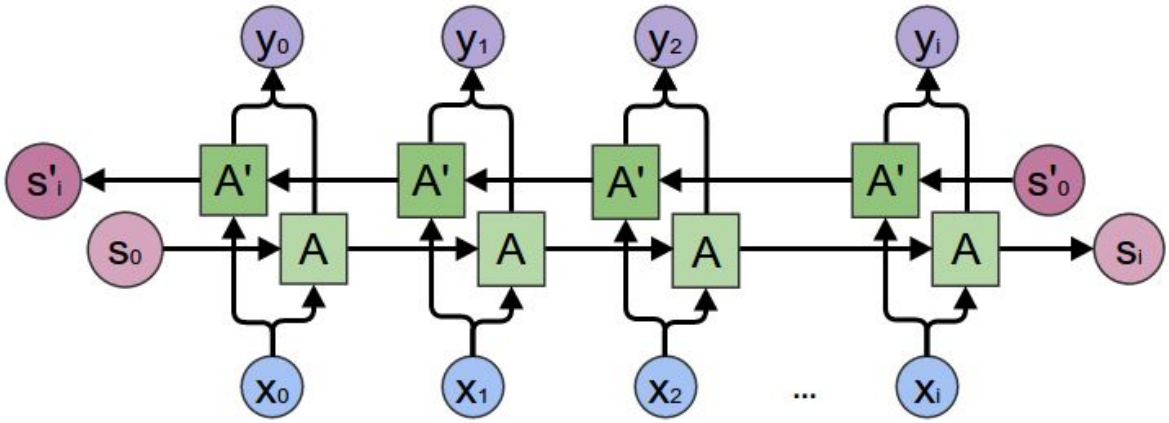

Figure: Bidirectional RNN

Figure Source: https://towardsdatascience.com/understanding-bidirectional-rnn-in-pytorch-5bd25a5dd66

Training

The loss function that we have used is a Cross-Entropy loss after using a softmax activation on the final layer outputs. We have used the Adam Optimizer to minimize this loss. The accuracy of Net1 is characterized by the percentage of phonemes correctly classified per frame. We evaluate on both the training and test set and plot the training and test error for each epoch.

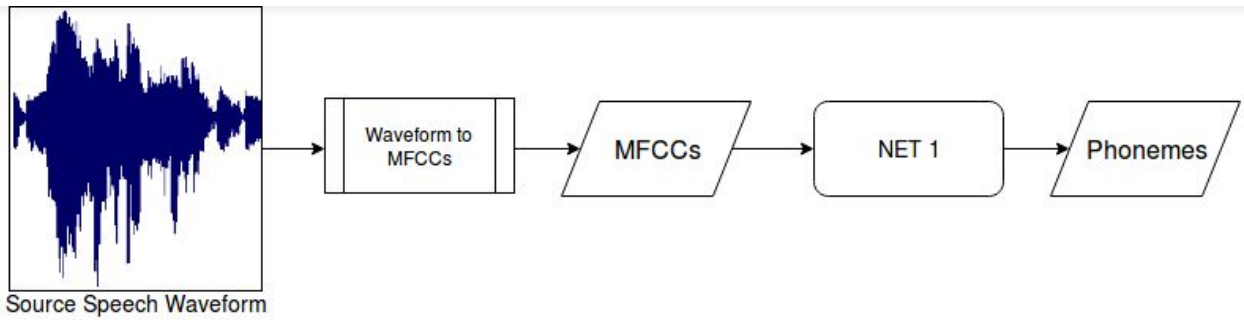

Figure: Stage 1 - Waveform to phonemes

Net2 - Phonemes to Target Speech

Dataset

For training Net2, we use the CMU Arctic dataset. This dataset has 593 train and 539 test samples of individual speakers of varying accents and genders which makes it a good dataset for training target voices. The phoneme label along with the end time (in secs) for each are available in the transcribed label files. We have trained our model on one of the speakers only. Half of the audio files were used as the training set and half were used as the test set.

While the TIMIT set uses the label *'h#'* for pauses, Arctic uses the label *'pau'*. Thus we had to convert the *'pau'* to *'h#'* in the arctic dataset.

Preparing the Input Data

In a manner similar to Net1, we create a batch using randomly selected 2-second samples. The phonemes per frame are represented by one-hot vectors. To feed the input we create an array of dimensions (BATCH_SIZE x TIME_FRAMES x 61), where BATCH_SIZE is the number of samples passed at once per batch, TIME_FRAMES is the number of frames per file and 61 is the number of phoneme classes.

Model Structure

The initial structure used for Net2 was similar to Net1. The output of Net2 is the log-magnitude of the STFT of the audio signal (257 values per frame). This is because reconstruction of an audio signal is better from the STFT than from the MFCCs.

The first structure that we tried was a bidirectional GRU network, like Net1. Next, we implemented a **pyramidal network structure** in which the number of units in each hidden layer increases from the input (61) to the output (257). The pyramidal structure gave a much lesser mean squared error compared to same number of units in each layer.

Multitask

Training a deep learning model end-to-end, *forces* the model to implicitly learn intermediate representations between the input and output. These states may have no meaning at all. For networks having meaningful intermediate states, domain knowledge can be leveraged to explicitly learn intermediate states. Using intermediate representations as auxiliary supervision can potentially give the best of both: end-to-end systems and pipelined approaches. [Ref: Toshniwal et al]. We've used the multitask approach to train the network as per both the mel spectral coefficients (as an intermediate representation) and magnitude spectrum which forms the final output.

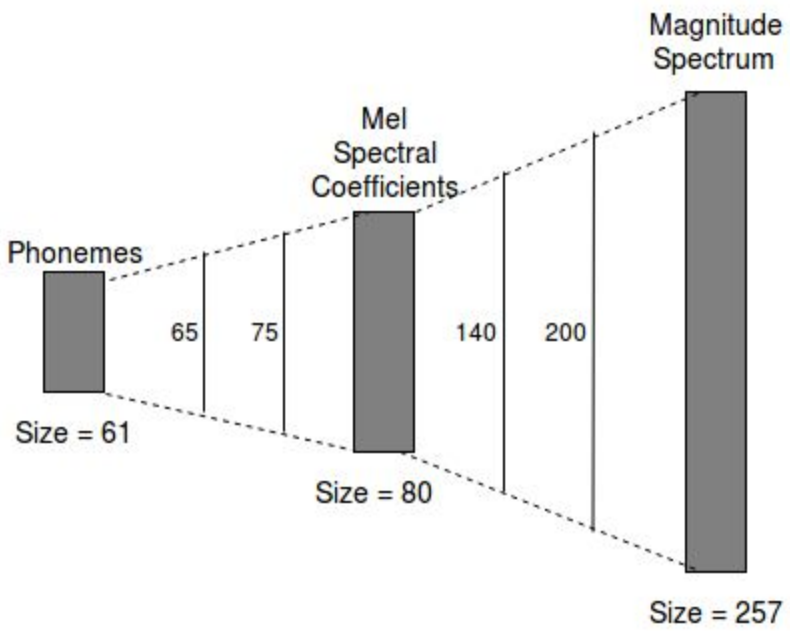

Figure: Multitask with pyramidal structure

Training

The loss metric that we have used is a Mean Squared Error between the predicted and actual magnitude spectrum values. For the multitask network, the loss function is the sum of mean squared errors of the mel spectrum and the magnitude spectrum. We have used the Adam Optimizer to minimize this loss. The accuracy of Net2 is indicates how off our predicted values are from the actual magnitude spectrum values. Akin to Net1, we evaluate on both the training and test set and plot the training and test error for each epoch.

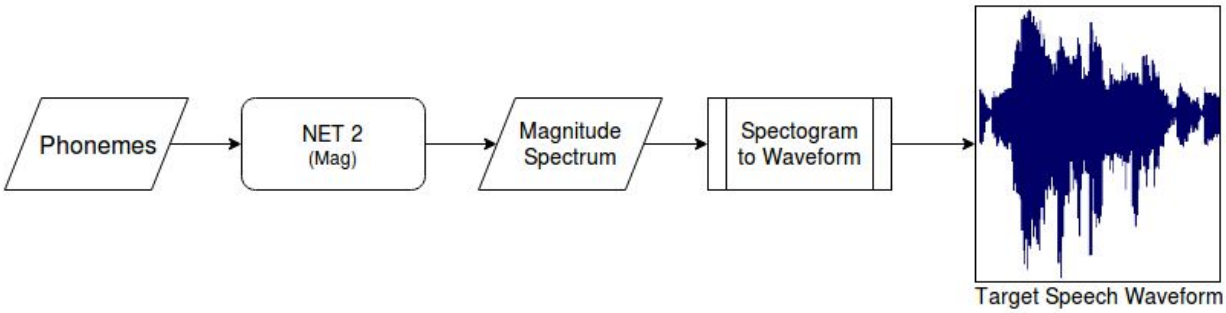

Figure: Stage 2 - Phonemes to waveform

Converting Magnitude Spectrogram to Waveform

We construct the target voice wav file from the magnitude waveform obtained as output of Net2. Since we only have the magnitude spectrum and no information about the phase, this reconstruction will not be perfect. For this, we use an iterative algorithm called the Griffin-Lim algorithm [Ref: Tacotron (Wang et al.)]. It begins with a random initialization of the phase spectrum. In every iteration, it computes the inverse-STFT, then re-estimates the phase of the STFT. When applied to the magnitude spectrum of a wav file itself, we found that the algorithm reconstructs the audio which sounds as good as the original.

Experiments and Results

We've performed extensive experimentation for both the neural networks varying hyperparameters including number of layers stacked in the LSTM cell, number of hidden units each cell emits, dropout probabilities as well as trying out different model structures.

The detailed results of all the experiments are provided in an appendix at the end of the report. A summary of the results is as follows:

Net 1-

The best accuracy we got on Net1 was just over 70%. This accuracy is a measure of the per frame phoneme classification. Net 1 does not output the actual phoneme sequence. For this reason, we haven't calculated the PER and have not benchmarked our results.

For predicting the phonemes from the input wav, Net1 does a decent job on clean American accent. More particularly, it performs well on the TIMIT and arctic test set. Quite a few of the misclassifications are due to predicting similar sounding (but different phonemes) in intermediate frames (for eg- '*ae*' instead of '*aa*').

For Net1, using LSTMs works better than using GRUs. This is owing to the ability of LSTMs to capture the link between the long chain of repeated phonemes. The best accuracy was obtained for a bidirectional LSTM with 2 hidden layers with 200 nodes each and dropout keep probability of 0.6.

Net 2-

Predicting Magnitude Spectrum: Bidirectional LSTM works out better in this case. Pyramidal network structure yields a marginal decrease in MSE.

Predicting Mel Spectrum and Magnitude Spectrum, multitask: Bidirectional GRU works out better in this case. Pyramidal network structure yields quite some decrease in MSE.

We found that the accuracy of Net1 didn't matter as significantly as the accuracy of Net2 did while converting a voice sample end-to-end. Many of the phoneme errors are to related phonemes and often a few phoneme errors in a chain of repeated phonemes does not

affect the audio much. On the other hand the spectrum prediction has a large impact on the quality of the converted speech.

End to End conversion-

We were able to successfully convert the source waveform to the target waveform retaining some features of the target speaker. The output voice succeeds in retaining the intelligibility of the spoken sentence, however it sounds unnatural and robotic.

Here is one example from the TIMIT test dataset (female voice) that was converted to a target voice (male) trained from the Arctic dataset:

Source Target

Future Work and Scope

Increasing the performance of Net2 will be the main focus going forward. We have found that varying hyperparameters does not increase the performance much. Directly predicting the complex STFT of the target audio is expected to give a better quality of the reconstructed audio than only predicting the magnitude spectrum.

We would like to explore and implement architectures used by leading models in Speech Generation like CBHGs used by Tacotron.

Another direction that we can go towards is implementing a Generative Adversarial Network (GAN).

Appendix : Detailed Report of Experiments

Net1

Unidirectional LSTM

Hidden Layers	Units per layer	Max. Test Accuracy (%)
1	50	68.3
	75	68.8
2	50	69.4
	75	70.0
	100	**71.1**
3	50	69.8
	75	70.2
	100	70.6
4	50	69.0
	75	**71.1**

Bidirectional LSTM

Hidden Layers	Units per layer	Keep probability	Max. Test Accuracy (%)
4	100	0.9	71.8
		0.8	72.5
		0.7	**72.8**
		0.6	72.3

Bidirectional GRU

Hidden Layers	Units per layer	Max. Test Accuracy (%)
1	50	67.9
	75	69.0
	100	69.4

2	50	69.9
	75	**71.1**
	100	70.3
3	50	70.4
	75	70.6
	100	70.8
4	50	70.7
	75	70.5

Bidirectional GRU and Bidirectional LSTM with dropout

Cell	Hidden Layers	Units per layer	Keep probability	Accuracy (%)
GRU	2	200	0.6	67.5
	3			62.2
	4			9.3 (outlier case)
LSTM	2	200	0.6	**73.5**
	3			70.5
	4			69.4

Net2

Bidirectional GRU

Hidden Layers	Units per layer	Keep probability	Mean Square Error
2	100	0.8	0.189
3			0.187
4			0.192
5			**0.185**

Bi-GRU with Pyramidal Structure

Units per layer	Keep probability	Mean Square Error
100, 150, 200	0.6	**0.181**
100, 150, 200	0.6	0.187
61, 124, 257	0.6	0.186

Bi-LSTM with Pyramidal Structure

Units per layer	Keep probability	Mean Square Error
100, 150, 200	0.6	**0.164**

Bi-GRU with Multitask (mags and mels)

Hidden Layers	Units per layer	Keep probability	Mean Square Error
2	100	0.6	0.383
3			0.391
4			0.388
5			0.392

Bi-GRU with Pyramidal Structure and Multitask (mags and mels)

Num. units for mags	Num. units for mels	Keep probability	Mean Square Error
65, 75	140, 200	0.8	0.359
65, 75	140, 200	0.6	**0.351**
100, 100	100, 100	0.9	0.384

Bi-LSTM with Pyramidal Structure and Multitask (mags and mels)

Num. units for mags	Num. units for mels	Keep probability	Mean Square Error
65, 75	140, 200	0.6	0.546