SpeedySpeech: Efficient Neural Speech Synthesis

Claudio Pastorini
University of Rome La Sapienza
pastorini.1792086@studenti.uniroma1.it

August 30, 2021

Contents

Chapter 1

Introduction

In this work we are going to reimplement the SpeedySpeech: Efficient Neural Speech Synthesis paper by Jan Vainer and Ondřej Dušek from Charles University, Faculty of Mathematics and Physics, Prague, Czechia.

In this paper the authors proposed a **Student-Teacher network** capable of **high-quality faster-than-real-time spectrogram synthesis**, with **low requirements on computational resources** and **fast training time**.
The results obtained seem to be surprising compared to the state-of-art solutions.

Chapter 2

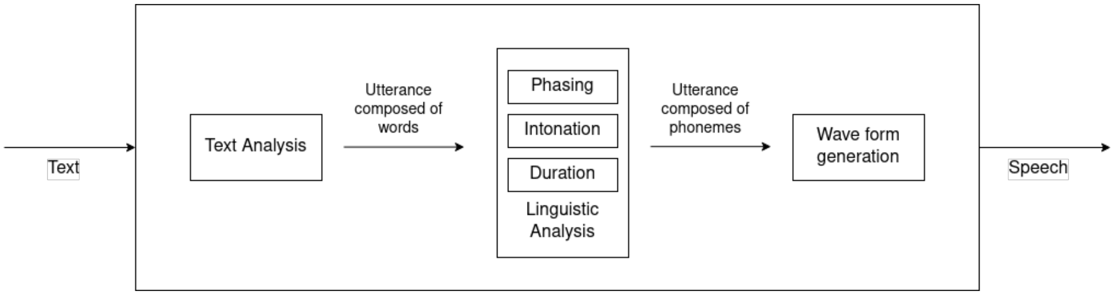
Text-To-Speech

First of all, we have to understand what a **Text-To-Speech (TTS)** system is.

From the Wikipedia page about Speech synthesis:

> Speech synthesis is the artificial production of human speech. A computer system used for this purpose is called a speech computer or speech synthesizer, and can be implemented in software or hardware products. A text-to-speech (TTS) system converts normal language text into speech; other systems render symbolic linguistic representations like phonetic transcriptions into speech.

A text-to-speech system is composed of two parts: a front-end and a back-end.



The front-end has two major tasks:

- First, it converts raw text containing *symbols like numbers and abbreviations into the equivalent of written-out words*. This process is often called **text normalization**, pre-processing, or tokenization.
- Then, it assigns *phonetic transcriptions to each word*, and *divides and marks the text into prosodic units*, like phrases, clauses, and sentences. The process of assigning phonetic transcriptions to words is called **text-to-phoneme** or grapheme-to-phoneme conversion.

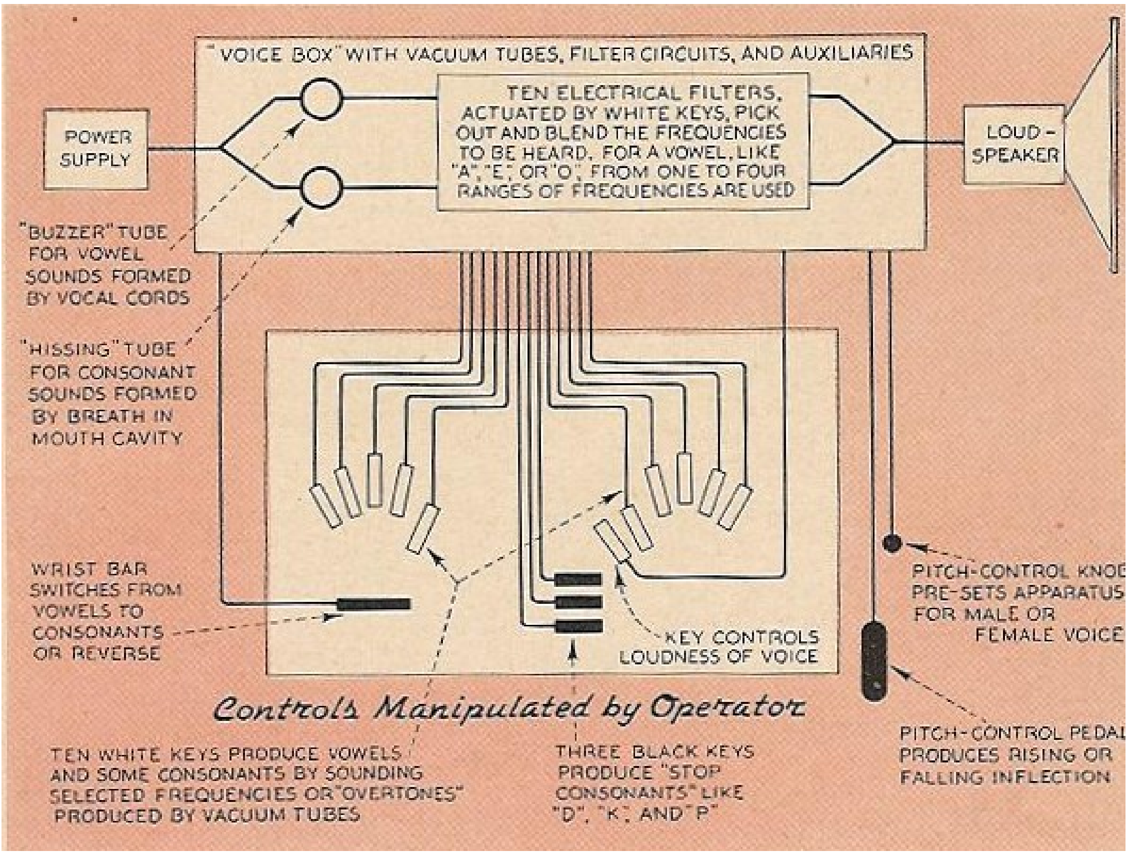
Phonetic transcriptions and prosody information together make up the symbolic linguistic representation that is output by the front-end.

The back-end — often referred to as the **synthesizer** — *converts the symbolic linguistic representation into sound*. In certain systems, this part includes the computation of the target prosody (pitch contour, phoneme durations), which is then imposed on the output speech.

2.0.1 The Voder

In the 1930s Bell Labs developed the vocoder, which automatically analyzed speech into its fundamental tones and resonances. From his work on the vocoder, Homer Dudley developed a keyboard-operated voice-synthesizer called **The Voder** (Voice Demonstrator), which he exhibited at the 1939 New York World's Fair.

The Bell Telephone Laboratory's Voder (Voice Operating Demonstrator) was the *first attempt to synthesize human speech by breaking it down into its component sounds and then reproducing the sound patterns electronically to create speech.*


"VOICE BOX" WITH VACUUM TUBES, FILTER CIRCUITS, AND AUXILIARIES
POWER SUPPLY
TEN ELECTRICAL FILTERS, ACTUATED BY WHITE KEYS, PICK OUT AND BLEND THE FREQUENCIES TO BE HEARD. FOR A VOWEL, LIKE "A", "E", OR "O", FROM ONE TO FOUR RANGES OF FREQUENCIES ARE USED
LOUD-SPEAKER
"BUZZER" TUBE FOR VOWEL SOUNDS FORMED BY VOCAL CORDS
"HISSING" TUBE FOR CONSONANT SOUNDS FORMED BY BREATH IN MOUTH CAVITY
WRIST BAR SWITCHES FROM VOWELS TO CONSONANTS OR REVERSE
KEY CONTROLS LOUDNESS OF VOICE
PITCH-CONTROL KNOB PRE-SETS APPARATUS FOR MALE OR FEMALE VOICE
Controls Manipulated by Operator
TEN WHITE KEYS PRODUCE VOWELS AND SOME CONSONANTS BY SOUNDING SELECTED FREQUENCIES OR "OVERTONES" PRODUCED BY VACUUM TUBES
THREE BLACK KEYS PRODUCE "STOP CONSONANTS" LIKE "D", "K", AND "P"
PITCH-CONTROL PEDAL PRODUCES RISING OR FALLING INFLECTION

2.0.2 Daisy bell

In 1961, physicist John Larry Kelly, Jr. and his colleague Louis Gerstman used an IBM 704 computer to synthesize speech, an event among the most prominent in the history of Bell Labs. Kelly's *voice recorder synthesizer (vocoder) recreated the song "Daisy Bell"*, with musical accompaniment from Max Mathews.

2.0.3 Deep learning-based synthesis

In September 2016, DeepMind proposed **WaveNet**, a *deep generative model of raw audio waveforms.* This showed the community that deep learning-based models have the capability to model raw waveforms and perform well on generating speech from acoustic features like spectrograms or spectrograms in mel scale, or even from some preprocessed linguistic features.

In early 2017, Mila (research institute) proposed **char2wav**, a *model to produce raw waveform in an end-to-end method.* Also, Google and Facebook proposed **Tacotron** and **VoiceLoop** respectively, to *generate acoustic features directly from the input text.* Later in the same year, Google proposed Tacotron2 which combined the WaveNet vocoder with the revised Tacotron architecture to perform end-to-end speech synthesis. Tacotron2 can generate high-quality speech approaching the human voice.

Since then, end-to-end methods became the hottest research topic because many researchers around the world started to notice the power of the end-to-end speech synthesizer.

2.1 Challenges

Different challenges were met by the researchers during the years. **Text normalization** is a known problem also in other fields such as Information Retrieval and several techniques were developed in order to address it.

Another problem is **text-to-phoneme**: there is a simple map between grapheme to phoneme but it is different for each language and there are different.

Another problem that raised with time is the problem related to the **evaluation** of the results generated by the synthesizer. With time the researcheds come to the conclusion that each work should be evaluated by a pool of users considering the audio generated by the synthesizer and other synthesizer with the same phrases.

A really big problem not yet issued is the one related to the **prosodics and emotional content**. The audio generated is always flat and without emotion. Using GAN seems to solve the problem addressing the prosodics, instead for the emotional content different models should be used trained on dataset with audios and emotion.

Chapter 3

Dataset

The dataset that will be used is the LJ-Speech-Dataset, a public domain speech dataset consisting of **13.100 short audio clips** of a single speaker reading passages from 7 non-fiction books.

A transcription is provided for each clip, whose length varies from 1 to 10 seconds, adding up to approximately 24 hours.

Metadata is provided in `transcripts.csv` file and it consists of one record per line, delimited by the pipe character (`0x7c`).

The fields are:

- **ID**: this is the name of the corresponding `.wav` file
- **Transcription**: words spoken by the reader (`UTF-8`)
- **Normalized Transcription**: transcription with numbers, ordinals, and monetary units expanded into full words (`UTF-8`).

Each audio file is a single-channel *16-bit PCM WAV* with a sample rate of *22.050 Hz*.

A brief example of the `transcript.csv`:

```
                                                Transcription  \
ID
LJ001-0001  Printing, in the only sense with which we are ...
LJ001-0002                     in being comparatively modern.
LJ001-0003  For although the Chinese took impressions from...
LJ001-0004  produced the block books, which were the immed...
LJ001-0005  the invention of movable metal letters in the ...

                                     Normalized Transcription
ID
LJ001-0001  Printing, in the only sense with which we are ...
LJ001-0002                     in being comparatively modern.
LJ001-0003  For although the Chinese took impressions from...
LJ001-0004  produced the block books, which were the immed...
LJ001-0005  the invention of movable metal letters in the ...
```

3.1 Graphemes vs phonemes

As for the speech synthesis, we are interested in the **phonemes** (*unit of sound*) and not **graphemes** (the *transcription of the phoneme*). So we need to transform all the graphemes, representing the transcription, into phonemes.

For this purpose, we will use the g2p library that can convert English graphemes to phonemes.

Using the library we are able to perform the transformation over all our samples and we obtain something like this:

```
                                             Transcription  \
ID
LJ001-0001  Printing, in the only sense with which we are ...
LJ001-0002                      in being comparatively modern.
LJ001-0003  For although the Chinese took impressions from...
LJ001-0004  produced the block books, which were the immed...
LJ001-0005  the invention of movable metal letters in the ...

                                  Normalized Transcription  \
ID
LJ001-0001  Printing, in the only sense with which we are ...
LJ001-0002                      in being comparatively modern.
LJ001-0003  For although the Chinese took impressions from...
LJ001-0004  produced the block books, which were the immed...
LJ001-0005  the invention of movable metal letters in the ...

                                                  Phonemes
ID
LJ001-0001  PRIH1NTIH0NG , IH0N DHAH0 OW1NLIY0 SEH1NS WIH1...
LJ001-0002  IH0N BIY1IH0NG KAH0MPEH1RAH0TIH0VLIY0 MAA1DER0N .
LJ001-0003  FAO1R AO2LDHOW1 DHAH0 CHAY0NIY1Z TUH1K IH0MPRE...
LJ001-0004  PRAH0DUW1ST DHAH0 BLAA1K BUH1KS , WIH1CH WER0 ...
LJ001-0005  DHAH0 IH0NVEH1NSHAH0N AH1V MUW1VAH0BAH0L MEH1T...


Pick sample: LJ008-0027.wav
```

And the metadata for this particular sample is:

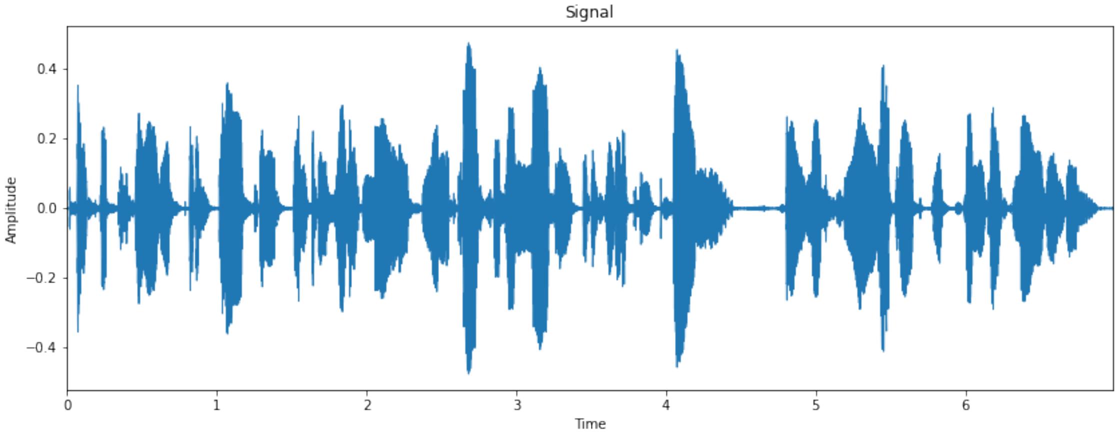
```
Transcription               put a quid of tobacco into his mouth, and offe...
Normalized Transcription    put a quid of tobacco into his mouth, and offe...
Phonemes                    PUH1T AH0 KWIH1D AH1V TAH0BAE1KOW2 IH0NTUW1 HH...
Name: LJ008-0027, dtype: object
```

3.2 Audio signal representation

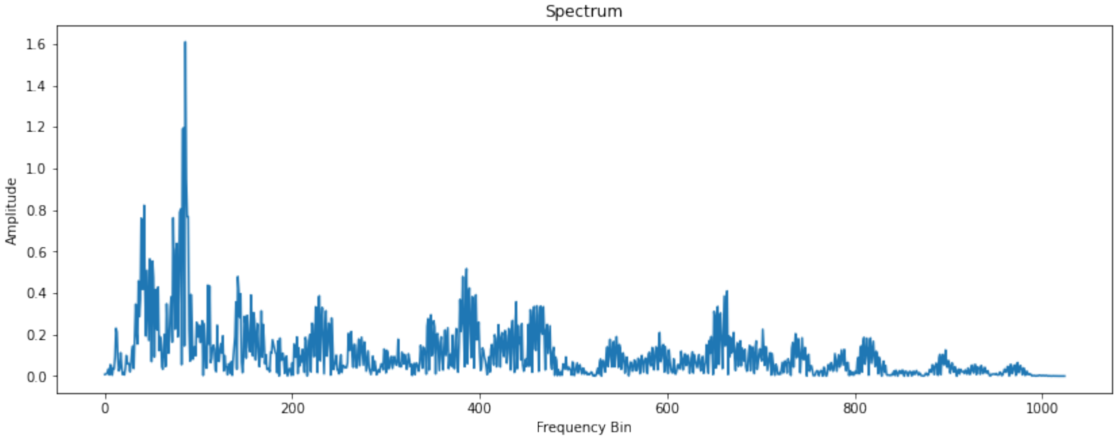
Before moving onto the network, it is important to understand all the different ways through which an audio signal can be represented.

One possible way is to simply represent the *variation of the amplitude along the time.*

So, considering the previous sample, we have the following representation:



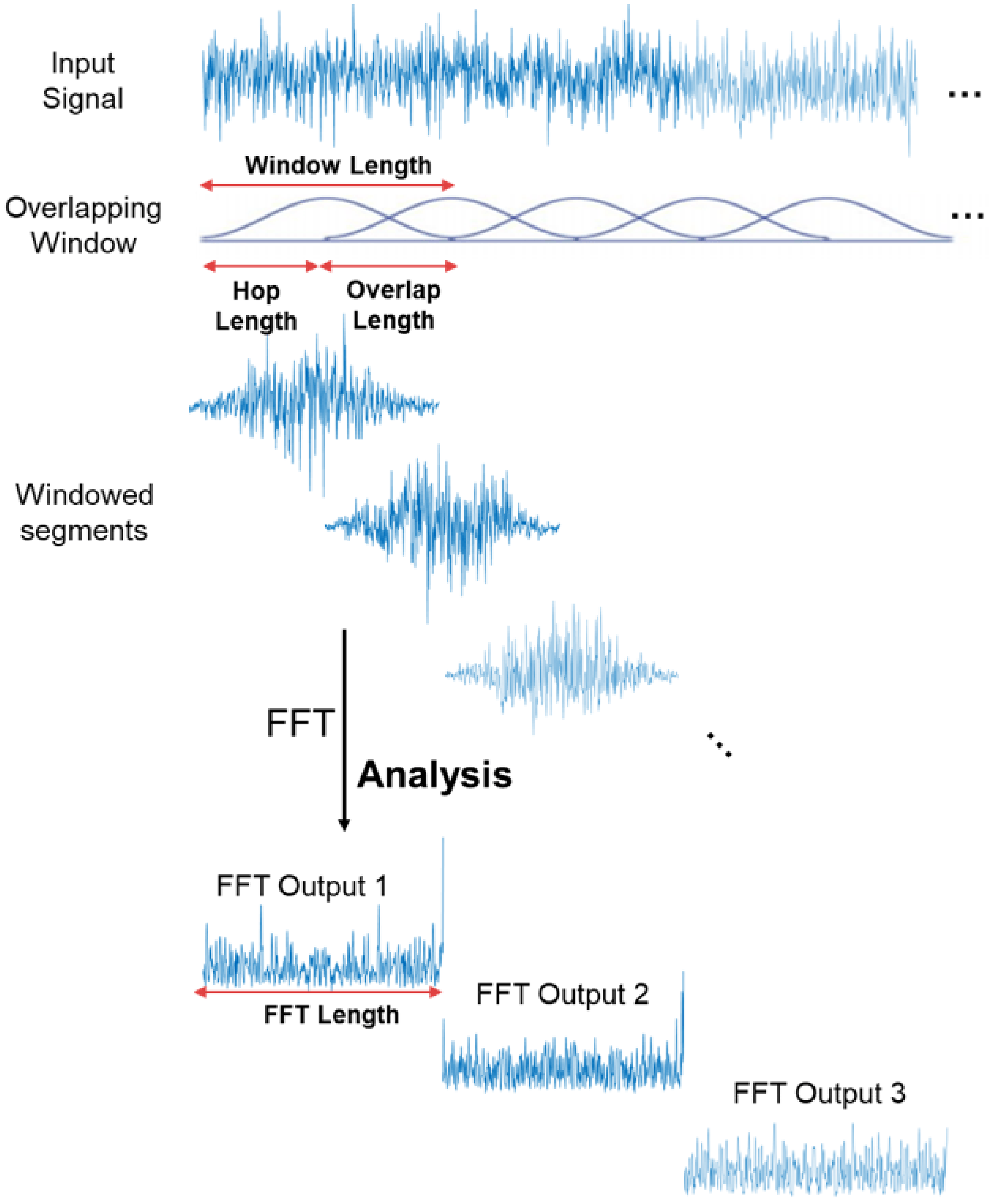
However, as each audio signal is composed by multiple frequencies, we can also represent it as a **spectrum**, so to show the *variation of the amplitude along the frequency* instead of time, thanks to the *Short-Time Fourier Transform algorithm*:



In this way we have lost the information about the time.

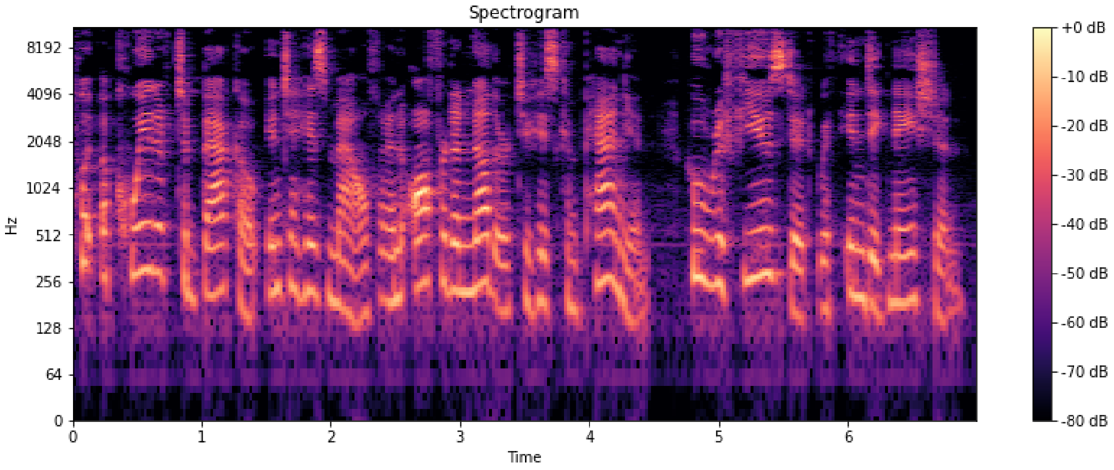
Generally, audio signals -such as music or voice- are not periodic; therefore, we lose an important piece of information by removing it from the graph.

A solution to this is to compute different *SFTs on overlapping windowed segments of the signal*, producing the so-called **spectrogram**.



With this representation we can see the *variations of frequency and amplitude during time* in our signal. Amplitude is expressed in dB with different colours.

The **spectogram** of our sample signal is:



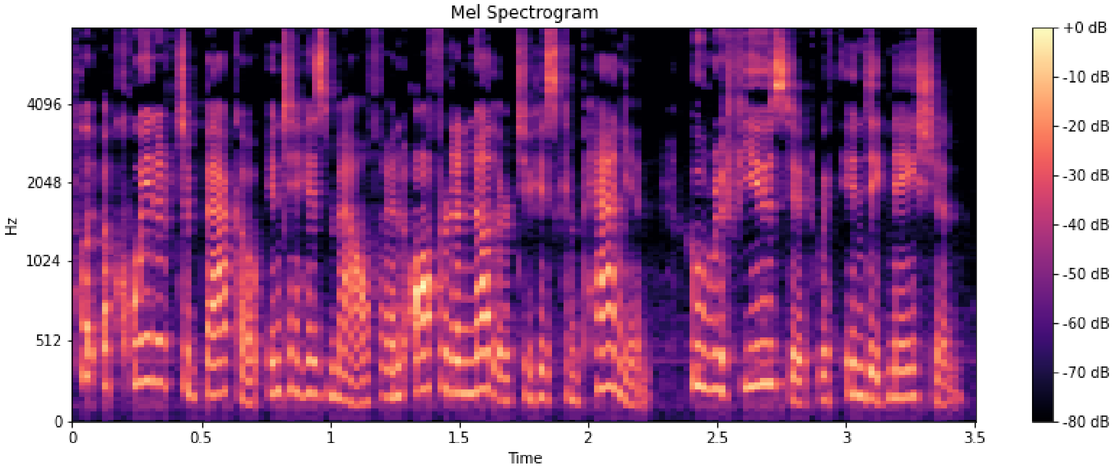
In this paper, as in others dealing with signals representing voices, the spectrogram used is not a normal spectogram but it the so-called **Mel spectogram**.

A Mel spectogram is a *spectogram that uses a different scale (the Mel scale) for the frequencies.*

The Mel scale was created by Stevens, Volkmann, and Newmann in the 1937 as a scale that *puts stress to the fact that humans are not able to identify well differences in higher frequencies.*

For example, we can easily tell the difference between 500 and 1000 Hz, but we will hardly be able to tell a difference between 10.000 and 10.500 Hz, even though the distance between the two pairs are the same!

So we need to remap the frequency scale to the Mel scale and we obtain the **Mel spectogram**:



Chapter 4

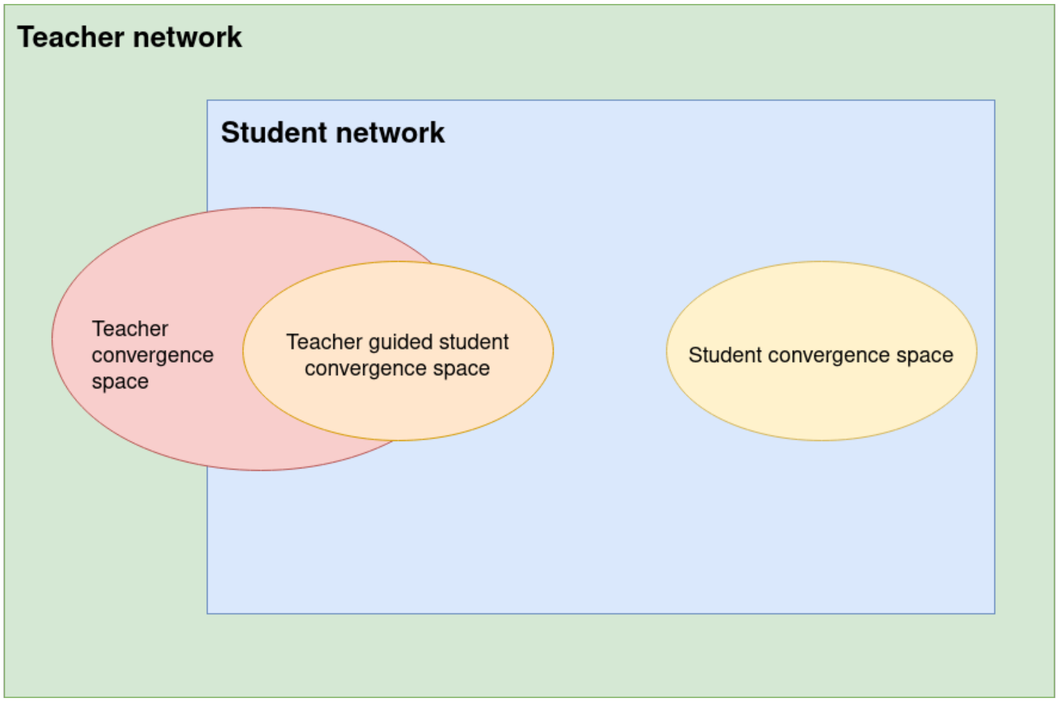
The network

4.1 The Teacher-Student network

This type of network was introduced to solve the problem of the overwhelming size of Deep Neural Networks (DNN). **Teacher-Student tries to transfer knowledge from a complex teacher network to a simple student network.**

The hypothesis is that the student will be able to learn the underlying concepts and knowledge learned by the teacher that the student otherwise wouldn't be able to learn because of its simpler architecture and fewer number of parameters. This knowledge transfer is achieved by minimizing the loss between the soft labels produced by the teacher and the student.

If the student network is guided to replicate the behavior of the teacher network (which has already searched through a bigger solution space), it is expected to have its convergence space overlapping with the original teacher network convergence space.

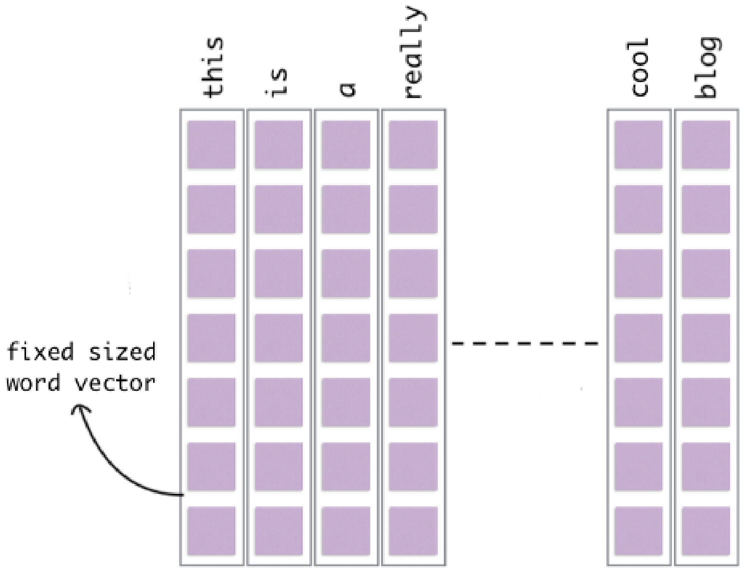

Teacher network
Student network
Teacher convergence space
Teacher guided student convergence space
Student convergence space

4.2 Layers

4.2.1 Embedding layer

It allows to **represent phonemes as dense vectors** where a vector represents the projection of the phonemes into a continuous vector space.

The position of a phoneme within the vector space is learned from the input and is based on the sorrunding phonemes.



At the creation the Tensor is initialised randomly. It is only after the train that the similarity between similar words should appear.

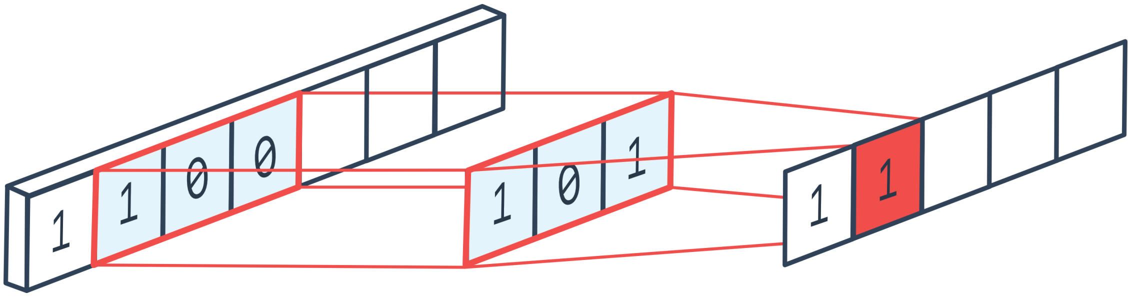
So, once an embedding layer is defined, and the vocabulary defined and encoded (i.e. assign a unique number to each word in the vocabulary) it is possible to use the layer to get the corresponding embedding.

This tecnique gives us a way to use an **efficient dense representation in which similar words have a similar encoding**. The encoding is not given by us but is learned by the network.

4.2.2 Fully connected layer

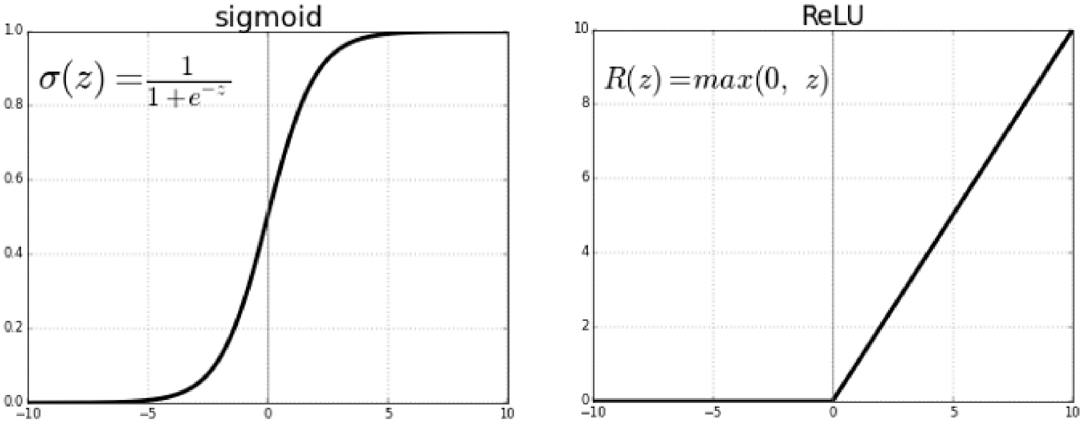
We are working with **audio data and text data**: both can be represented as **time series data that use a single dimension**.

For this reason we will use **one dimensional convolutional layer with kernel slides along one dimension**.



4.2.3 Activation functions

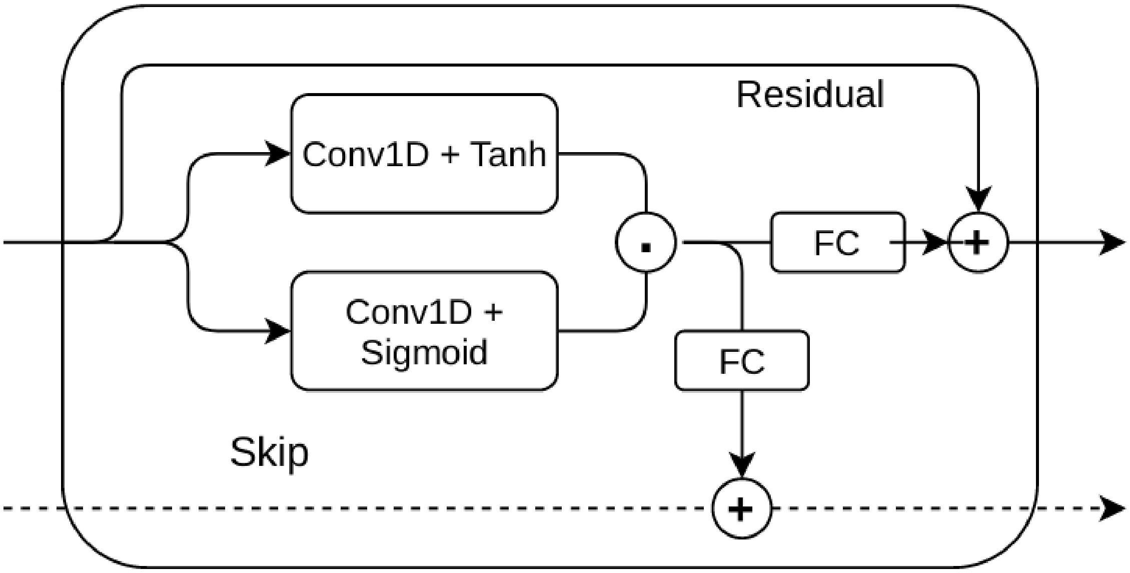
Rectified Linear Unit (ReLU) is used for most of the network. For the last step of the Teacher network a **Sigmoid** activation function is used, whereas for the Student network an **Identity** activation function is used.



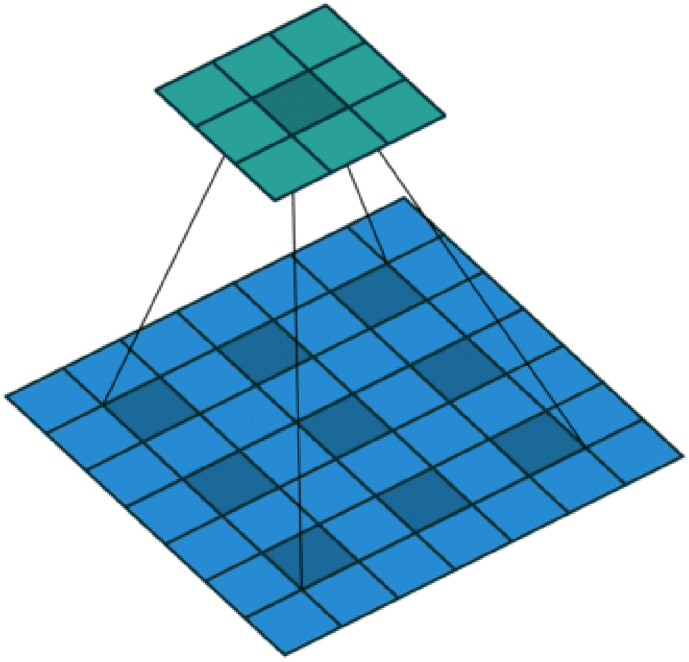
4.2.4 Wave Residual Block

Residual block was **taken by WaveNet**. **It captures dependency between the data points at distance** thanks to the dilation layers.

It produces two outputs: 1. Feature map from the previous residual block, which will be used as an input for the next one. 2. Skip connection, which will be used to calculate loss function for the batch after the aggregation.



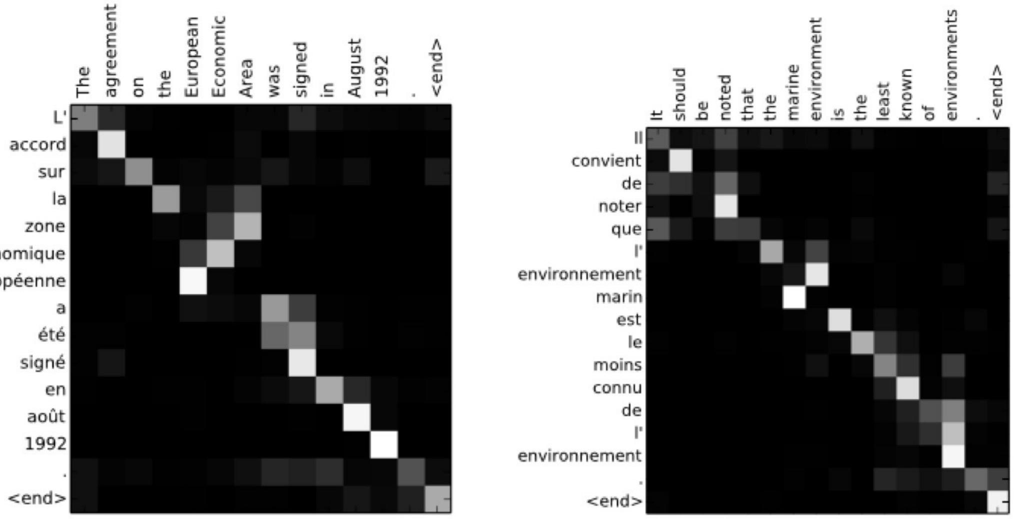
A **dilated convolution** is a convolution where the filter is applied over an area larger than its length by **skipping input values with a certain step**. It is equivalent to a convolution with a larger filter derived from the original filter by dilating it with zeros, but is significantly more efficient.



4.2.5 Attention

A transformer is an architecture composed by an encoder, an attention function and a decoder that is able to take care of previous inputs much better than RNN and LSTM. In this case, it is able to **consider the relative importance of a spectogram with respect to an input sequence of phonemes**.

An attention **function can be described as mapping a query and a set of key-value pairs to an output**, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key.



4.3 Loss functions

In the paper the authors use the **sum of L1 loss between the target and predicted spectrograms and guided attention loss for the attention module** in the Teacher network.

On the other hand, for the Student network, they use the **sum of L1 loss and structural similarity index (SSIM) losses for logarithmic Mel spectrogram value** and **Huber loss for logarithmic duration prediction**.

4.3.1 L1 loss

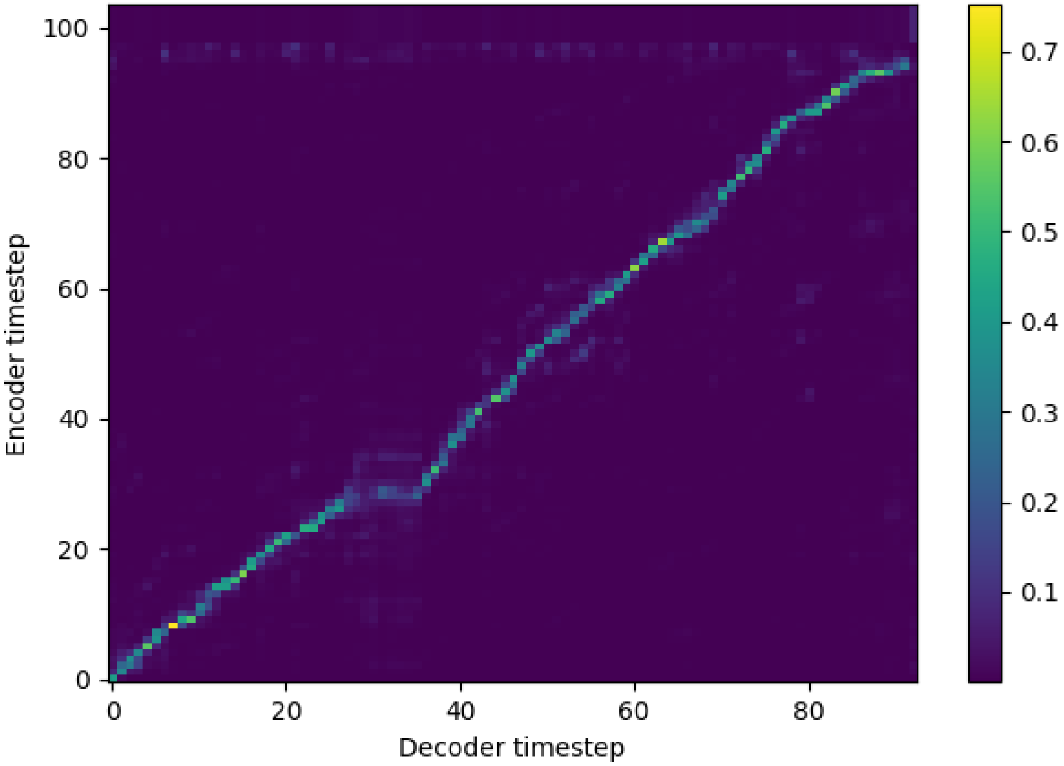
L1 loss is the **Mean Absolute Error (MAE)**. MAE is the sum of absolute differences between our target and predicted variables. It measures the average magnitude of errors in a set of predictions, without considering their directions.

4.3.2 Guided Attention Loss

Generally, an **attention module is quite costly to train**. Therefore, if there is some prior knowledge, incorporating them into the model may be a help to alleviate the heavy training.

So this type of loss prompts the **attention matrix to be "nearly diagonal", speeding up the training**.

In TTS, the possible attention matrix lies in the very small subspace because of the **rough correspondence of the order of the characters and the audio segments**. That is, when one reads a text, it is natural to assume that the text position progresses. This is a noticeable difference of TTS from other seq2seq learning techniques such as machine translation, where an attention module needs to resolve the word alignment between two languages that have very different syntaxes, e.g. English and Japanese.



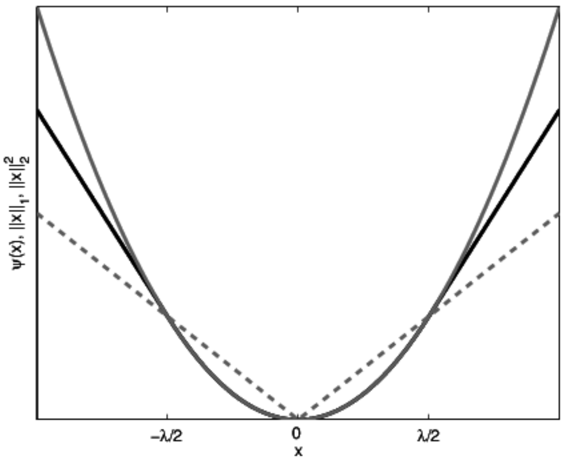
4.3.3 Structural Similarity Index Measure (SSIM)

The **Structural Similarity Index Measure (SSIM)** is a method for measuring the **similarity between two images**. The SSIM index can be viewed as a quality measure of one of the images being compared, provided the other image is regarded as of perfect quality.

Here it is used to evaluate the **similarity of the spectograms generated**.

4.3.4 Huber loss

The **Huber Loss is a combination of the L1 and L2 functions**. It is **quadratic for small values and linear for larger values**. It combines the best features of both error functions keeping it differetiable.



4.4 Optimizer

Both the networks use the **Adam optimization algorithm**. The Teacher uses a **Noam scheduler** while the Student network uses a **Reduce Learning Rate on Plateau**.

4.4.1 Adam

The **Adam optimization algorithm** is an **extension to stochastic gradient descent** that has recently seen broader adoption for deep learning applications in computer vision and natural language processing.

The method computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients.

Adam **combines the best properties of the AdaGrad and RMSProp algorithms** to provide an optimization algorithm that can handle sparse gradients on noisy problems. Adam is relatively easy to configure where the default configuration parameters do well on most problems.

4.4.2 Noam Scheduler

The Noam scheduler **increases the learning rate linearly** for the **first so-called warm-up steps**, and **decreases it thereafter** proportionally to the **inverse square root of the step number**.

4.4.3 Reduce Learning Rate on Plateau

In general this scheduler **reduces learning rate when a metric has stopped improving**. In this case, it is when the sum of all the validation losses stops to improve.

4.5 Teacher network

The Teacher network is used for **extracting phoneme durations from data**.

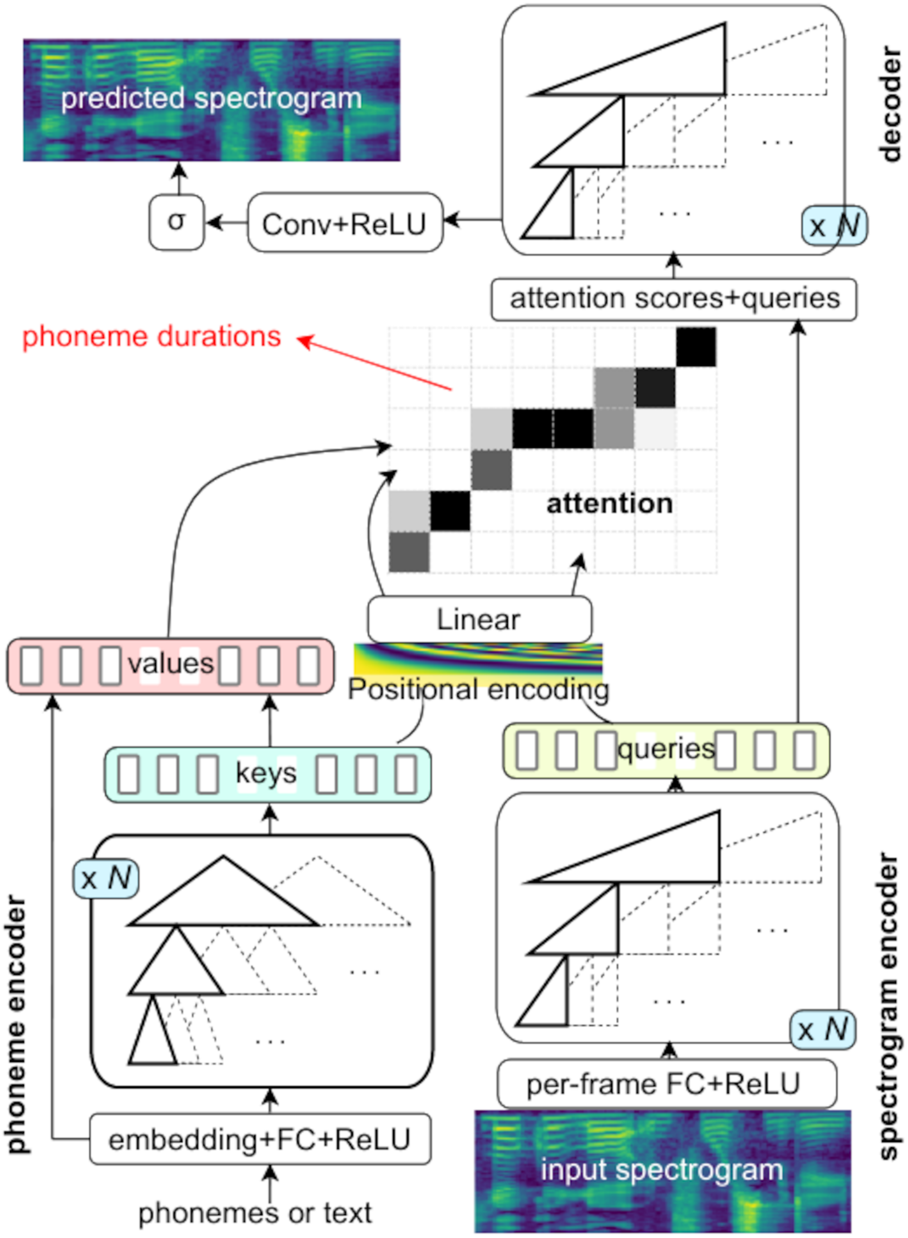
It is composed by:

- phoneme encoder
- spectrogram encoder

- attention
- decoder

It is trained to **predict the following spectrogram frame, given input phonemes (including punctuation) and past frames**; it uses attention to keep track of the phoneme it is generating.

The **attention values** are then **used to align phonemes with spectrogram frames and extract phoneme durations**.



4.5.1 Phoneme encoder

The phoneme encoder takes **phonemes** as **input** and **produces the keys and values** to be used with the **attention layer**.

It is composed of:

- an embedding layer
- fully connected layers
- ReLU as activation function
- several Wave Residual blocks

4.5.2 Spectogram encoder

The spectogram encoder takes **the spectogram** as **input** and **produces the queries values** to be used with the **attention layer**.

It is composed of:

- fully connected layers
- ReLU as activation function
- several Wave Residual blocks

4.5.3 Attention

The attention layer takes **the keys, values and the queries** as **input** then it **produces attention scores on phonemes durations**.

It is composed of:

- attention layer
- fully connected layers

4.5.4 Decoder

The decoder takes **attentions scores and queries** as **input** and then **generates predicted spectrograms**.

It is composed of:

- several Wave Residual blocks
- fully connected layers
- ReLU as activation function
- Sigmoid as last activation function

After the training of the Teacher model, we are ready to extract the durations to feed them inside the Student network.

We load the last checkpoint of the model and we run it against the whole dataset producing a `.csv` file.

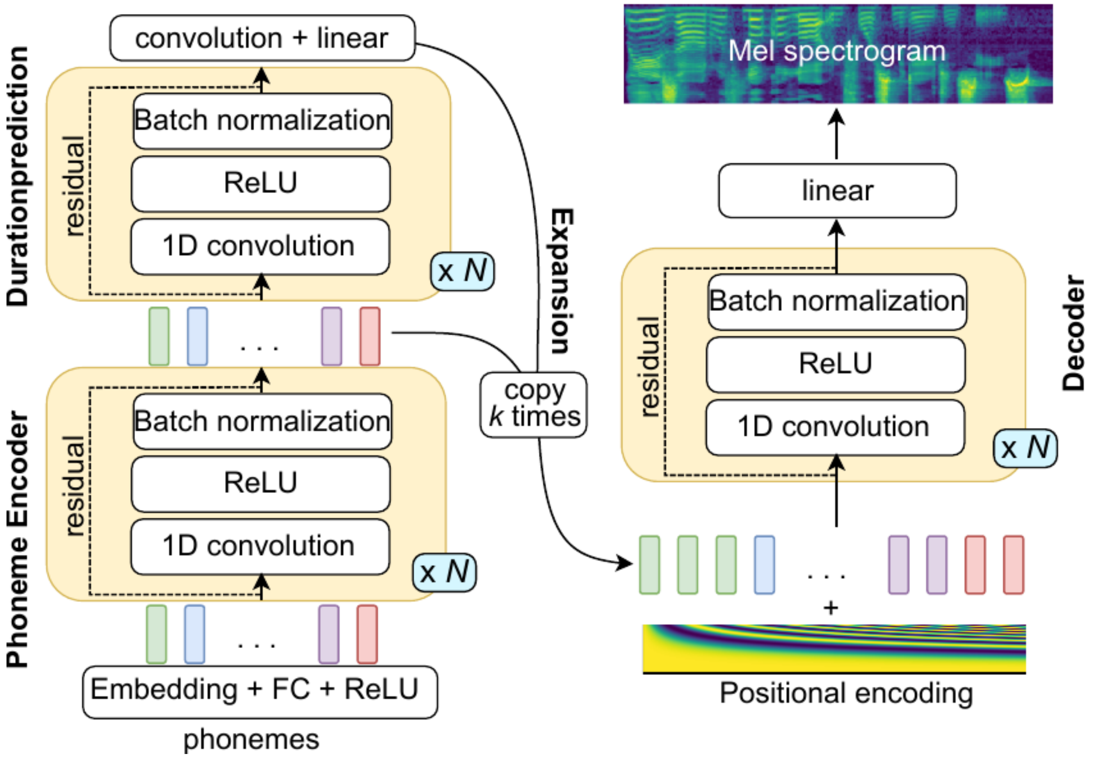
The generated file contains all the extracted durations to be used in the next phase.

Here follows a sample:

4.6 Student network

The student network **will produce the Mel spectogram** representing the synthetized voice and it is composed of:

- phoneme encoder
- duration predictor
- decoder



4.6.1 Phoneme encoder

The phoneme encoder takes **phonemes** as **input** and **generates data** to be fed to the duration predictor and decoder.

It is composed of:

- an embedding layer
- fully connected layers
- ReLU as activation function
- several Wave Residual blocks

4.6.2 Duration predictor

The duration predictor takes **the phonemes encoded** by the previous layer **as input** and, **guided by the durations** of phonems generated by the Teacher network, it will **produce phonemes durations**.

It is composed of:

- several Wave Residual blocks
- fully connected layers

4.6.3 Decoder

The decoder takes the **phonemes and durations as input** and **generates the Mel spectogram**.

It is composed of:

- several Wave Residual blocks
- fully connected layers
- identity as activation function

Chapter 5

Result

The training was made using the Paperspace service through a **P4000** instance that has the following specifications:

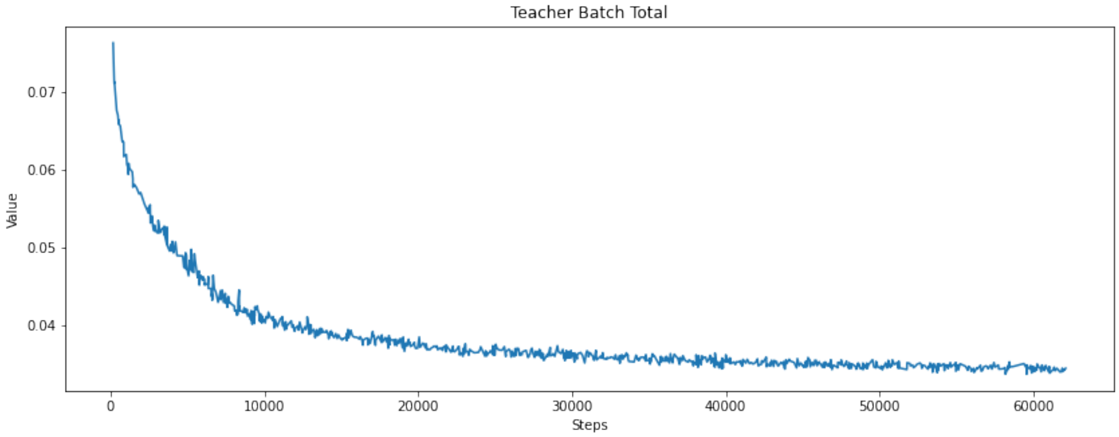
- Intel Xeon 4215 with 8 cores, 3.2 GHz (max turbo frequency of 4 GHz)
- GPU+ Gen 2 (P4000) with 8 GB GDDR5 (1,792 CUDA cores) (5.3 TFLOPS)
- 30 GB RAM

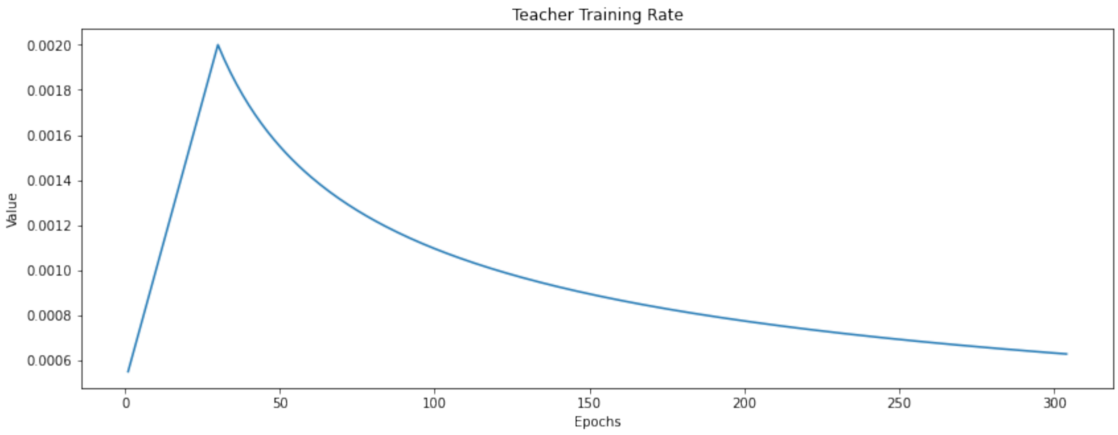
The training of the **Teacher model** needed **20 hours and 35 minutes**, while the training of the **Student model** needed **11 hours and 28 minutes**.

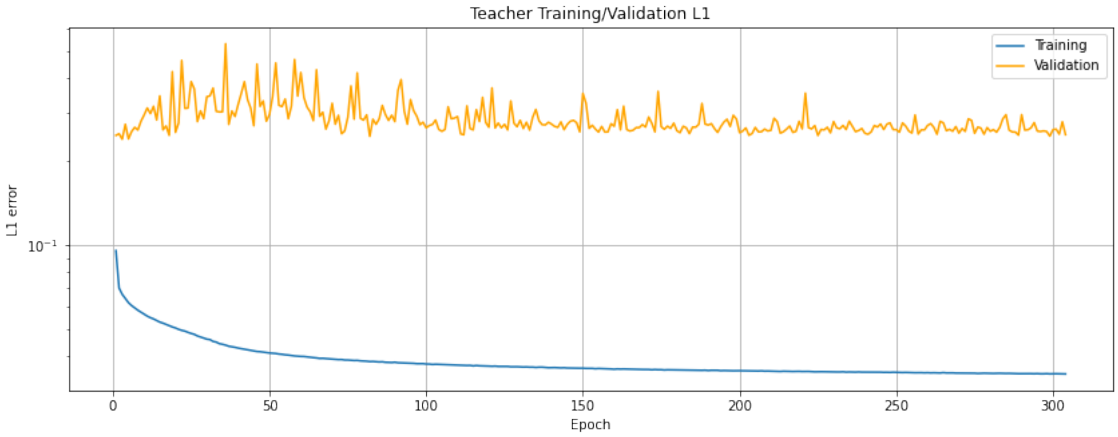
The total time needed for the training of the entire dataset was **32 hours circa**.

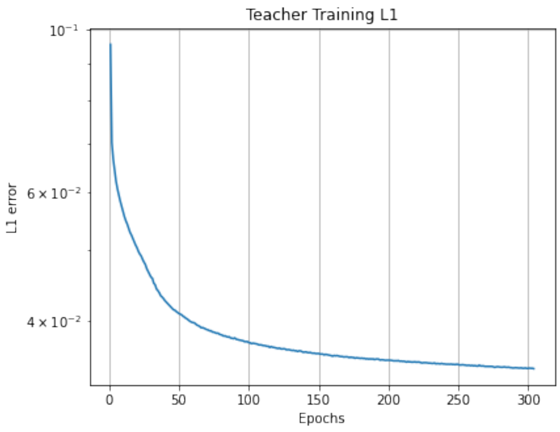
A comparable time with the original one that stated 32 hours (19 for the teacher and 13 for the student) with a similar computing power (a single GeForce GTX 1080 GPU with of 8GB RAM).

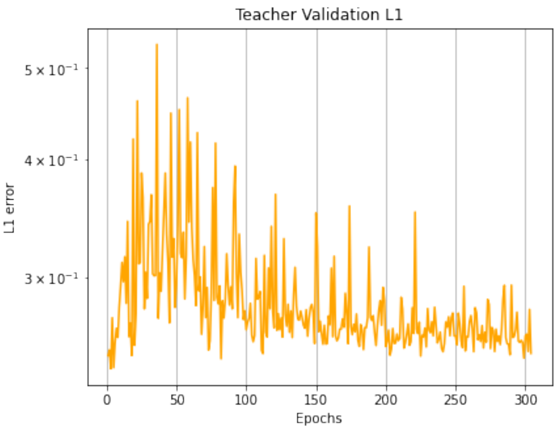
Even if the **Teacher model** is smaller, it takes longer to train it - circa **20 hours** - since a smaller learning rate must be used to converge with good results. The **Student model** takes circa **11 hours**. This is due to the fact that the architecture is simpler and does not contain any hard-to-train components such as attention, which makes it converge easier.

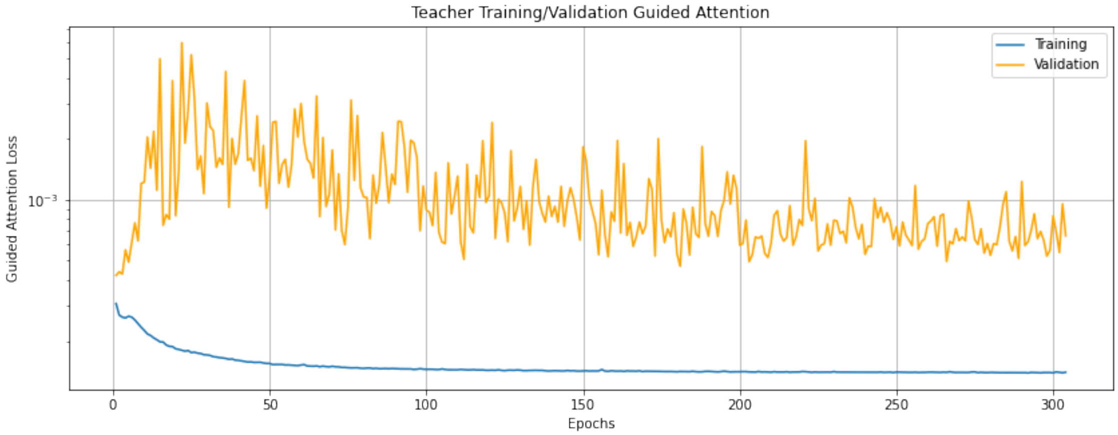


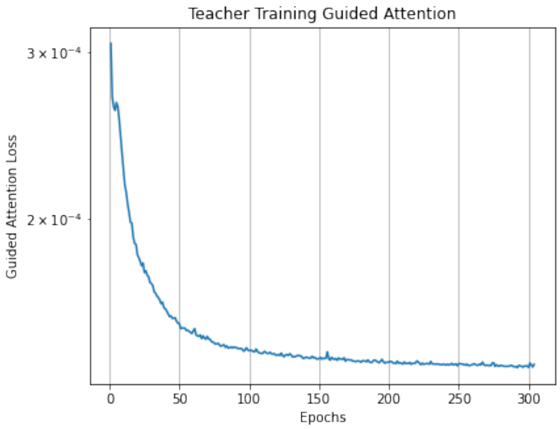

Teacher Training Rate
Value
0.0020
0.0018
0.0016
0.0014
0.0012
0.0010
0.0008
0.0006
0
50
100
150
200
250
300
Epochs

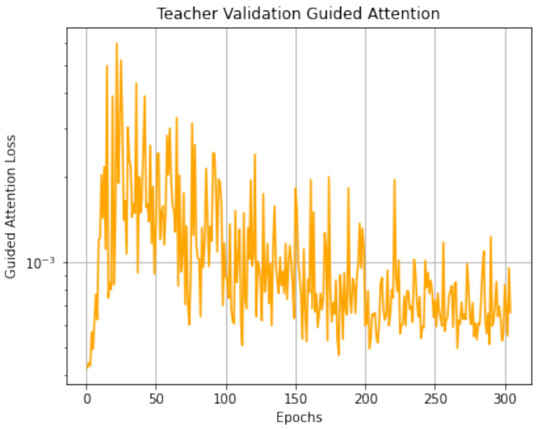

Teacher Training/Validation L1
Training
Validation
L1 error
10^{-1}
0
50
100
150
200
250
300
Epoch

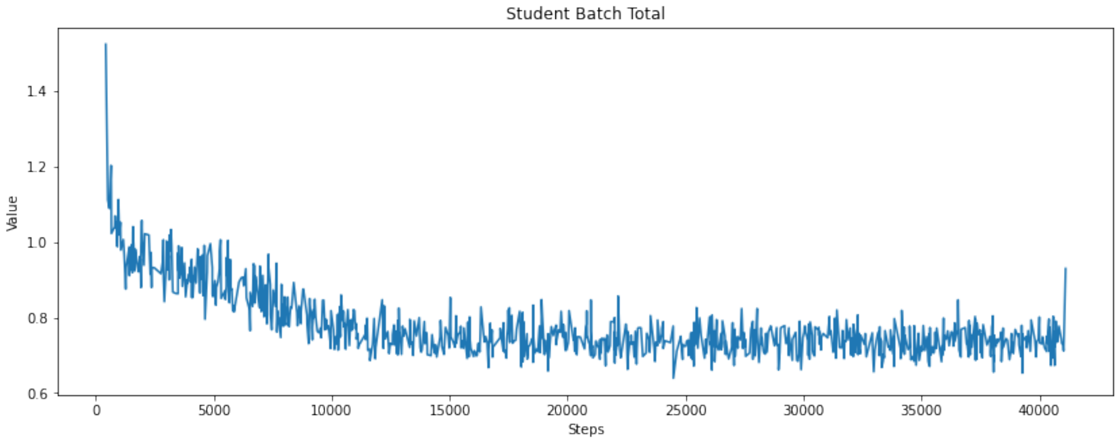

Teacher Training L1
L1 error
10^{-1}
6×10^{-2}
4×10^{-2}
0
50
100
150
200
250
300
Epochs

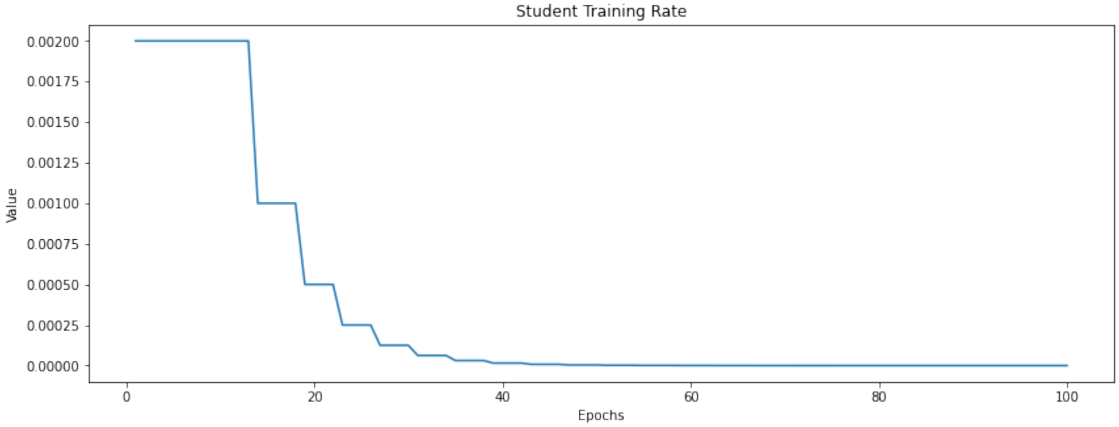

Teacher Validation L1
L1 error
5×10^{-1}
4×10^{-1}
3×10^{-1}
0
50
100
150
200
250
300
Epochs

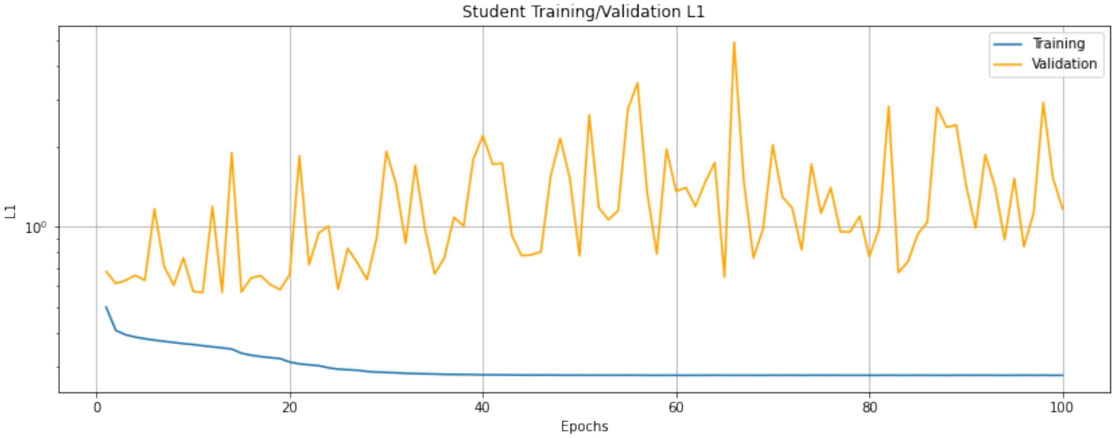

Teacher Training/Validation Guided Attention
Training
Validation
Guided Attention Loss
10^{-3}
0
50
100
150
200
250
300
Epochs

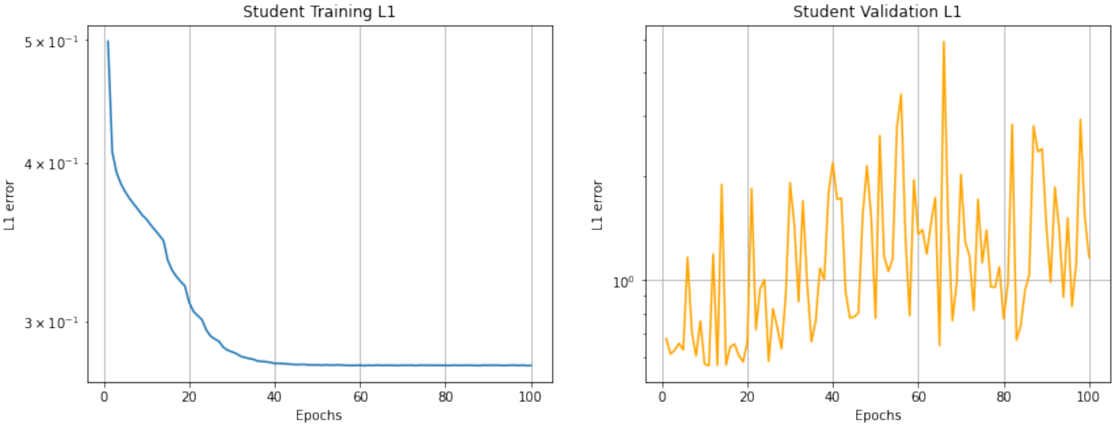

Teacher Training Guided Attention
Guided Attention Loss
3×10^{-4}
2×10^{-4}
0
50
100
150
200
250
300
Epochs

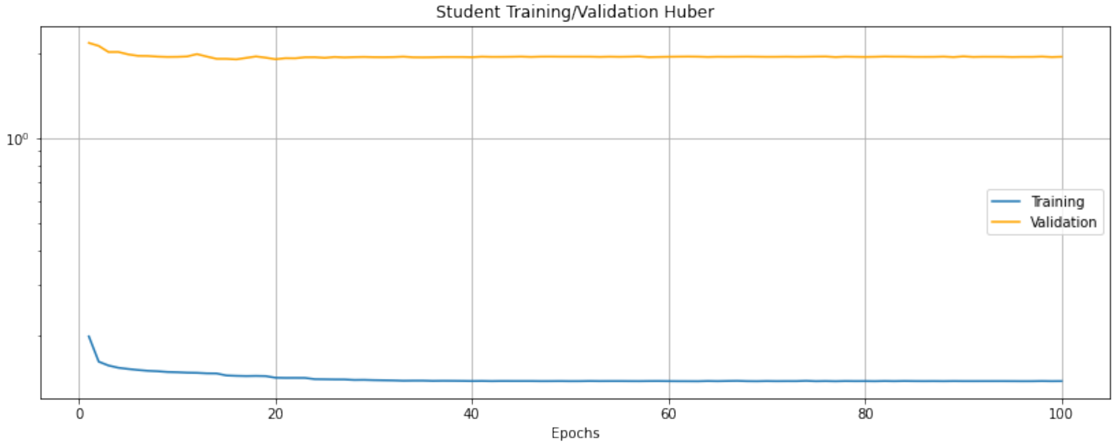

Teacher Validation Guided Attention
Guided Attention Loss
10^{-3}
0
50
100
150
200
250
300
Epochs

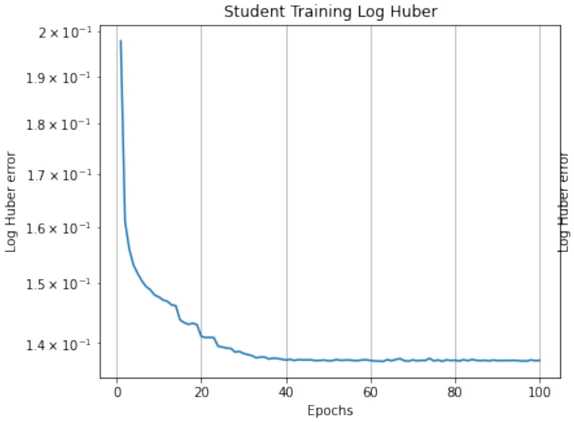

Student Batch Total
Value
1.4
1.2
1.0
0.8
0.6
0
5000
10000
15000
20000
25000
30000
35000
40000
Steps

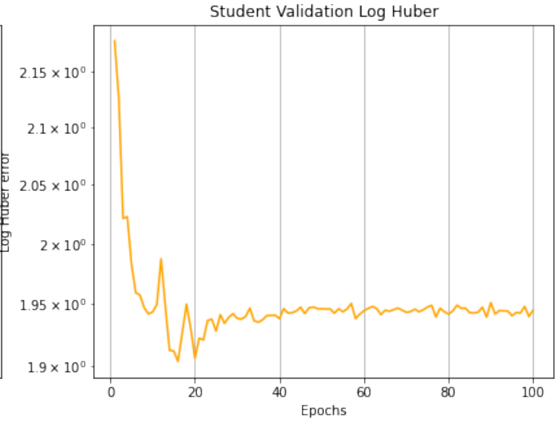

Student Training Rate
Value
0.00200
0.00175
0.00150
0.00125
0.00100
0.00075
0.00050
0.00025
0.00000
0
20
40
60
80
100
Epochs


Student Training/Validation L1
Training
Validation
L1
10^0
0
20
40
60
80
100
Epochs


Student Training L1
L1 error
5×10^{-1}
4×10^{-1}
3×10^{-1}
0
20
40
60
80
100
Epochs
Student Validation L1
L1 error
10^0
0
20
40
60
80
100
Epochs







As previously stated, it is difficult to evaluate the quality of the voice, for this reason the authors conducted an in-house survey with 40 participants.

They selected audio examples produced by the following setups for comparison:

- Reference human audio recording
- Deep Voice 3 + lws vocoding
- Tacotron 2 + MelGAN vocoding
- This solution + Griffin-Lim vocoding
- This solution + MelGAN vocoding

In particular the authors makes the survey using a MUSHRA based test. The participants were shown anonymized outputs of all models and the reference for a given sentence, and they rated them on a finegrained 100-point scale, visually divided into 5 categories: “Excellent”, “Fair”, “Good”, “Poor” and “Bad”.

Here the results of the MUSHRA-like scores from the survey, with 95% confidence intervals:

Model (vocoding)	Mean Score	95 % CI
Tacotron 2 (MelGAN)	62.82	(–2.01, +2.20)
Deep Voice 3 (lws)	43.61	(–2.25, +2.20)
Reference	97.85	(–0.76, +0.66)
Ours (Griffin-Lim)	47.03	(–2.00, +2.16)
Ours (MelGAN)	**75.24**	(–1.91, +1.73)