RhySpeech: A Deployable Rhythmic Text-to-Speech Based on Feed-Forward Transformer for Reading Disabilities

Yixuan Lin
YK Pao School
sailerneeter@qq.com

ABSTRACT

Dyslexia was first proposed in 1877, but this century-old problem still troubles many people today [1]. Dyslexia is marked by difficulty in reading despite having normal or superior conditions in their environment and intellectual ability, is curable using multi-sensory learning, which involves providing audio stimulus, sometimes generated from expressive text-to-speech. However, such generated audio lacks rhythmic features, marked by inadequate insertion of pauses. In response to such technological difficulty, this paper proposes RhySpeech, which models rhythm using feed-forward transformer neural networks and an LRV (Latent Rhythm Vector). The LRV receives input from the pitch, energy, and duration features encoded using a Transformers network along with the numeric encoding of the previous 16 phonemes, which together build a strong sense of context for the pause prediction. This LRV is trained to generate adequate lengths and positions of pa uses, allowing the synthesized audio to have more accurate pausing

CCS CONCEPTS

• **Applied computing** → Life and medical sciences; Bioinformatics.

KEYWORDS

Text-to-Speech, Deployable, Rhythmic, Transformer, Latent Vector

ACM Reference Format:
Yixuan Lin. 2023. RhySpeech: A Deployable Rhythmic Text-to-Speech Based on Feed-Forward Transformer for Reading Disabilities. In *2023 2nd Asia Conference on Algorithms, Computing and Machine Learning (CACML 2023), March 17–19, 2023, Shanghai, China.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3590003.3590062

1 INTRODUCTION

Human speech consists of sentences, units of speech representing relatively complete and independent meaning; which can be broken down into words, which have their definition, and humans tend to pause between words. By definition, a text-to-speech process is a digital process that takes text as input, then, through multiple steps, converts the text into sound waves (which resemble human speech) which are outputted. Sometimes, the term ”end-to-end” is added as a modifier to emphasize that the system is responsible for the entire process of text-to-speech.

Modern end-to-end text-to-speech systems consist of three steps (which text goes through to become audio) as shown in figure: text analysis, acoustic modeling, and the vocoder, which will be explained below in figure 1.

Text analysis involves converting input text (to-be-read) into a format that the acoustic model can accept as input. Usually, this step involves the conversion of numbers and symbols, which exists in some texts such as textbooks and science books, into their word form (which is what words will be spoken if a person is to read that number or symbol out loud). However, the spelling of a word doesn't carry a one-to-one relationship about how the word should be pronounced. For example, the ”h” grapheme in ”hill” is pronounced with the ”h” sound, but the same ”h” grapheme in ”honor” is not pronounced. To solve this issue, a grapheme-to-phoneme (G2P) conversion is carried out. Either a G2P dictionary or a G2P conversion model is used for the process.

The acoustic model is the part of text-to-speech that converts phonemes to acoustic features, or indicators of how text is to be pronounced, complete with details such as volume and sound frequency. Variance encoders, which are popular approaches to capturing key features of pronouncing phonemes, are situated in this step. Variance encoders operate by extracting a specific feature of audio from the training set that varies with the phoneme involved (such as the volume of speech which can be extracted from the amplitude of the sound wave) and training the feature of the sound wave in association to the corresponding phoneme. Variance encoders are usually trained using machine learning, where their parameters will adjust according to training data. An alternative approach to modeling core characteristics of audio is GAN, which is short for generative adversarial networks. A GAN involves a generator trying to generate audio as close to a human voice as possible, while the discriminator tries to tell apart the generator's artificial audio from a real human's speech. Through training, the generator and discriminator both improve in abilities, thus generating increasingly realistic speech audio.

The vocoder is the final part of text-to-speech that converts acoustic features to sound waves, what humans perceive as speech. Modern vocoders tend to use neural networks as the approach to speech synthesis.

Recent studies on text-to-speech can be divided into autoregressive and non-autoregressive approaches. In the autoregressive approach, speech generation models utilize computational results and byproducts as a basis for future computation, therefore requiring the model to construct the final speech audio linearly, from the start to the end of the text for which the speech needs to be generated. The inability to perform generative computations in

CACML 2023, March 17–19, 2023, Shanghai, China

ACM ISBN 978-1-4503-9944-9/23/03…$15.00
https://doi.org/10.1145/3590003.3590062

Figure 1: A pipeline adumbrating the general text-to-speech process.

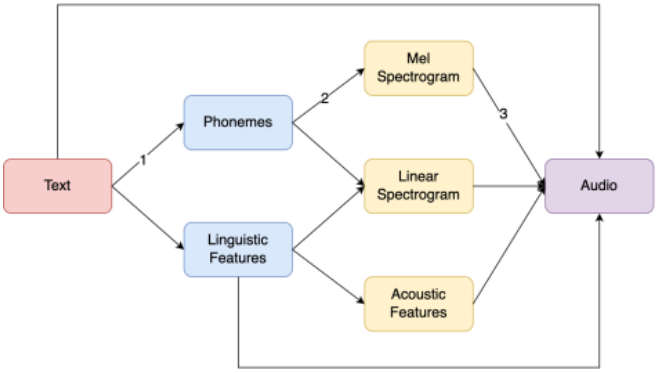


Figure 2: Variations in the process from text to speech [2]. Note that the pathway from the red to blue boxes is the phoneme encoding process, from the blue to the yellow boxes is the acoustic model, and from the blue to the purple box is the vocoder. The research follows a pathway labeled 1, 2, and 3 (from text to phonemes, to meet spectrogram, and finally to audio).

parallel impairs its processing speed, which is a significant letdown for text-to-speech applications where responses must be generated in very short times. In an attempt to reduce the computation time, the non-autoregressive approach does not rely on previous computations to make future computations, instead using other methods to ensure the quality of the generated speech, allowing parallel construction of speech audio, thus decreasing the generative latency of speech. The high potential speeds that non-autoregressive models process makes it the center of attention for researchers, believing that the non-autoregressive approaches have large potential for development. Due to the advantage of fast processing speeds, this research also uses the non-autoregressive approach.

Also, approaches to the text-to-speech task can be classified according to intermediate steps of speech synthesis. As shown in Figure 2, each approach selectively passes through steps of acoustic and linguistic feature analysis. This research follows a pathway from text to phonemes, mel spectrogram, and finally to audio.

Upon the problem of speech synthesis systems having inadequate rhythmic prediction capabilities under a small model size, this research proposes RhySpeech, a non-auto-regressive text-to-speech system based on Fastspeech2. RhySpeech specially features a pause predictor based on the transformer architecture of the audio's pitch and energy values. The training of the system is enhanced by RP-Sigmoid, a trainable activation function. The contributions of this research can be summarized as two main points:

1. Pause prediction via transformer architecture's input
2. RPSigmoid, a randomly initialized and parametric activation function based on the sigmoidal curve in figure 2.

2 METHOD

The overall process for text-to-speech in this research will be introduced in the section. Figure 3 shows the process of the inference stage when text is converted to speech, and Figure 4 shows the process of training.

2.1 Input Text Processing

The processing of inputted text involves breaking it down into individual phonemes. For this research, this is done using cmudict, which creates a long dictionary of words and their respective phoneme sequence, which is an ordered set of smallest units of distinct sound. Each phoneme sequence is described with a space inserted between each phoneme, and the vowels are marked with numbers to indicate the amount of stress that should be applied to them: 0 means there should be no stress applied, 1 means there should be primary (strongest) stress applied, and 2 means there should be secondary (next-strongest) stress applied. Note that the phonemes words are broken up into can be spelled differently from the original word since different spellings can have the same sound in English, thus making them all belong to the same phoneme. The same word, if labeled with 1 and 2 beside them, can be pronounced differently, such as presents primary stress is on the first "e" if the word is a noun, and on the second "e" if the word is a verb.

The processing of inputted text involves breaking it down into individual phonemes. For this research, this is done using cmudict, which creates a long dictionary of words and their respective phoneme sequence, which is an ordered set of smallest units of distinct sound. Each phoneme sequence is described with a space inserted between each phoneme, and the vowels are marked with numbers to indicate the amount of stress that should be applied to them: 0 means there should be no stress applied, 1 means there should be primary (strongest) stress applied, and 2 means there should be secondary (next-strongest) stress applied. Note that the phonemes words are broken up into can be spelled differently from the original word since different spellings can have the same sound in English, thus making them all belong to the same phoneme. The same word, if labeled with 1 and 2 beside them, can be pronounced differently, such as presents primary stress is on the first 'e' if the word is a noun, and on the second 'e' if the word is a verb.

Next, the phonemes are encoded into vectors for ease of storage because the storage of character sequences as characteristics of words is memory intensive and unable to be handled by neural networks, which only accept numerical inputs.

The last step of the preprocessing of data is to reduce the dimensionality of the vector. Aiming to use the available computational resource wisely, the vectors are run through an encoder algorithm, outputting more space-efficient vectors.

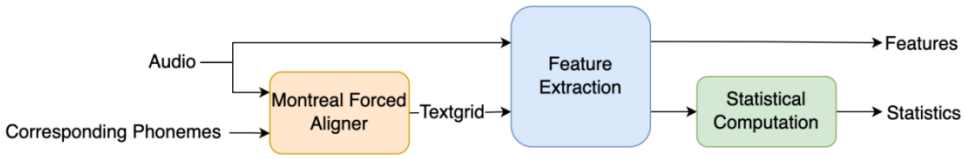


Figure 3: This diagram shows how the audio samples are preprocessed to obtain the features which are used as the ground truth by the acoustic model. Corresponding phonemes are also inputted to assist with the process of ground-truth-extraction.

Table 1: A selection of words from cmudict. The words are broken up into phoneme sequences. Screenshot from cmudict, extracted from https://raw.githubusercontent.com/ cmusphinx/cmudict/master/cmudict.dict

Word (grapheme)	Phoneme (individual sounds that make up the word)
Ablate	AH2 B L EY1 T
Ablation	AH2 B L EY1 SH AH0 N
Ablaze	AH0 B L EY1 Z
Able-Bodied	EY1 B AH0 L B AA1 D IY0 D
Abled	EY1 B AH0 L D
Abler	EY1 B AH0 L ER0
Abler(2)	EY B L ER0

2.2 Audio Preprocessing

The audio preprocessing stage obtains the ground truth values for the later acoustic model by preparing and extracting useful information from the speech recording, which are the attributes of pitch, duration, and energy, along with the mel spectrogram. A diagram for the audio preprocessing process is included in Figure 5.

The inputted audio recorded from an English speaker is fed into the system as .wav files, where the sound wave is saved as floating point numbers, indicating the height of the wave at that instant.

To accommodate readers with color vision differences, figures should still be usable when printed in grayscale. Refer to elements of the figure with non-color terms, for example “indicated as squares” instead of “indicated in blue”. Use different patterns in bar charts, different line patterns in graphs, and different shapes in plots to distinguish groups of elements and reinforce color differences.

2.2.1 Montreal Forced Aligner. The audio waveform along with the corresponding phonemes (produced by the input text processing) are passed into the Montreal Forced Aligner (MFA) [3], which marks the duration of each phoneme by referring to the audio waveform.

The MFA is an open-source model pretrained on the Speech dataset (with approximately 1000 hours of English speech). Pretraining follows a regime of 40 iterations of monophonic training, which by considering phonemes one at a time in alignment, then 35 iterations of triphone training, which considers one phoneme before and after the to-be-aligned phoneme. The monophonic training phrase aims to offer the model general alignment capabilities, and the triphone training fine-tunes the model by requiring it to consider the context. Realignment is performed on 20 (out of 40) iterations during monophonic training and 15 (out of 35) during triphone training, allowing the model to improve upon itself.

On a closer scale, the MFA model is trained with the acoustic feature of mel-frequency cepstral coefficients (MFCCs), which represent the intensity of different meet frequencies (frequencies with equal perceptual distance to humans, which puts it roughly on a log scale). The MFCCs are then put through Cepstral Mean and Variance Normalization (CMVN) to obtain a zero-median and unit-standard-deviation distribution to improve robustness.

2.2.2 Feature Extraction. The audio information is also passed along with the textgrid into the feature extraction, which finds certain characteristics of the audio inputted.

The first component of the extraction process is the mel spectrogram constructor. A mel spectrogram is an image that can represent the features of the audio. Its x-axis denotes the time of the audio, its y-axis is the log of the human-perceive frequency, and the color of a given pixel is rendered based on the intensity of that specific frequency at that specific timing.

To perform this construction, Short-Term Fourier Transform (STFT) is first applied to the audio recordings. The transformation process involves breaking the audio recording into short frames of around 20 to 50 milliseconds. The process only works through these short frames because human speech, on such a small scale, is relatively regular and predictable. Short-Term Fourier Transform utilizes this regularity to break the complex sound wave in each frame into compositions of sinusoidal waves at varying frequencies and varying amplitudes (intensities). Algebraically, this is represented as:

$$fk = \frac{1}{2\pi} \int dx f(x)\, e^{-ikx}$$

where f is the result for x’s Fourier Transform and k is the wavenumber.

To tackle the human pitch-determining inaccuracy, where humans typically perceive sounds higher than 1000 Hz lower than their actual pitch, a mel spectrogram adjusts for this hearing subjectivity, making it reflect what humans actually perceive.

The pitch is computed using the dio tool from Pyworld, which receives the .wav audio file as input and output the fundamental frequency (the loudest pitch at a specific time, which is the one that listeners hear and interpret) of audio at a specific time.

The energy is computed alongside the mel spectrogram when the Short-Term Fourier Transform represents the audio in an instant as differing intensities of differing frequencies. The energy captures this intensity.

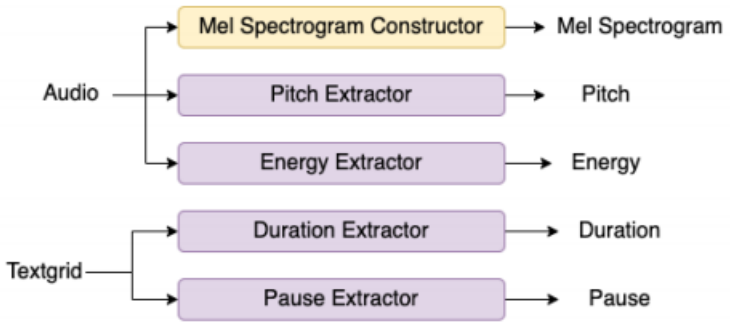


Figure 4: This diagram shows the extraction process used to create the ground truth predictions from the textgrid and the audio.

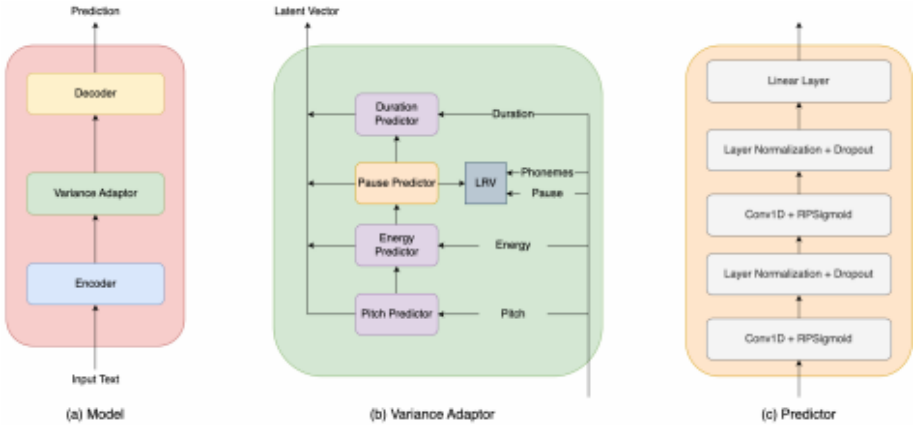


Figure 5: This diagram shows the acoustic model and its components used in this research

The duration, along with the usage of pauses in human speech, is extracted from the textgrid. Since the textgrid holds the start and end time of each phoneme and silence, such time values can directly be used as the duration and pause values. Specifically, if the timings on the textgrid correspond to an uttered phoneme, that timing length is labeled to the duration of the corresponding phoneme; if the timings correspond to silence, that timing length is labeled to the pause after the phoneme before it. If there is no silence after a phoneme, the pause information for that phoneme is labeled as 0. The last phoneme's pausing is also labeled as 0 since there is no need to save silence after the last phoneme because that silence doesn't improve audio quality but increases the file size in figure 5.

2.2.3 Statistical Computation. The values for pitch and energy are standardized using the StandardScalar() of scikit-learn [4]. First, the mean and standard deviation (measuring how spread-out the data is, relative to the center of it) are computed using built-in functions of the StandardScalar. Note that the standard deviation can be computed using the equation.

$$\sigma = \sqrt{\frac{\sum (x_i - \mu)^2}{N}}$$

where σ represents the standard deviation of the data, xi is the ith value in the data, μ is the mean of the data, and N is the number of elements in the data.

Then, the values of pitch and energy are transformed (using specific operations while not distorting the relative data) into a normal distribution: a median of 0, and a standard deviation of 1. This is achieved through a formula of

$$z = \frac{x - u}{s}$$

where z is the output of the standard scalar, x is the input, u is the mean, and s is the standard deviation. This allows for pitch and energy to be compared "equally", so that the influence of the exact value of pitch and energy, confounding variables, are removed.

Additionally, the minimum and maximum values for pitch and energy are computed. This would serve the purpose of establishing bounds in later stages of the research in functions such as rendering mel spectrograms.

2.3 Acoustic Model

Aiming to introduce abilities of expression and naturalness to the generated audio, this research project holds the ability for the audio-generation system to learn duration, energy, and pitch by building on Fastspeech 2; and additionally allowing the system to learn the rhythm. The framework for the acoustic model is shown in Figure 5

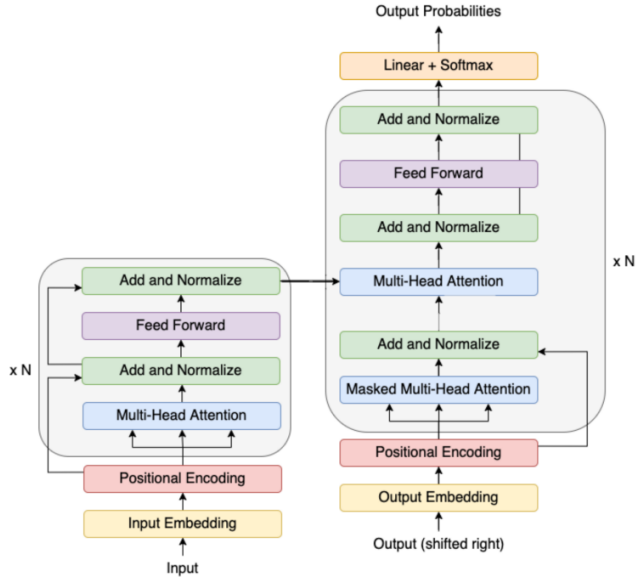


Figure 6: This diagram shows the structure of the transformer neural network

2.3.1 Transformer Architecture. The acoustic model uses a Transformer neural network architecture [5], as shown in Figure 6. The usage of a Transformer network allows the acoustic model to predict of features in multiple levels of focus, helping the vocoder access more detailed and accurate information about the acoustic features of the audio to be generated.

The Transformer structure includes an encoder, a variance adaptor (the name comes from its ability to learn the differing nuances in speech audio), and a decoder. The encoder is responsible for converting information into vector form useful for the future. The variance adaptor learns the variance present in speech. The decoder converts information from the encoded vectors back into normal values

Note that there could be multiple layers of encoder and decoder components to increase the model's capabilities. This means multiple degrees of content can be reframed into vectors, allowing the variance adaptor to receive better quality information.

The encoder and decoder share a similar architecture, which is the Transformer architecture, as shown in Figure 6.

To start with, the vectors entering the Transformer neural network are embedded, or converted, into vectors. This is because neural networks can only deal with groups of numbers (namely, vectors).

Afterward, positional encoding allows each element in the group of data to have a unique real number to indicate its position in the dataset. This allows for position to be effectively represented and considered in later steps, which is useful for time-dependent data analysis. The positional encoding process uses the formula of

$$PE \parallel mod_2\,(n) = 0 : \sin\left(\frac{n}{1000^{\frac{n}{d}}}\right)$$

$$PE \parallel mod_2\,(n) = 1 : \cos\left(\frac{n-1}{1000^{\frac{n-1}{d}}}\right)$$

where PE represents the result of positional encoding, n is the index of the to-be-positionally-encoded data, and d is the dimensions in the vector.

Such a method of positional encoding brings the benefit of not belittling other encoded data (if using the index to encode, the index might get much larger than other non-positional-encoding values, therefore making the model only focus on positional encoding values). This method also attributes similar weighting to the position regardless of data vector size, so that the difference of one element apart sets the positional encoding values to a similar difference.

Afterward, a multi-head attention operation is applied to the values passed along, regardless of coming from the "input" or "output" channel. Attention networks, in general, have the formula of

$$y = soft\max\left(\frac{QK^T}{\sqrt{d_k}}V\right)$$

where y is the output of the attention network, Q, K, and V are vectors from the data, and dk is the number of dimensions of the data vector K . The division by $\sqrt{dk}$ prevents dimensionality from changing the variance in the data, removing dimensionality as a confounding variable in the analysis. In multi-head attention, Q,K,V come from different places.

For the "output" channel, values undergo a masking process before this multi-head attention process. Iterating through the data, for every element, those elements later than it will be hidden (or masked), so that during the later attention mechanism, any later elements cannot be referenced (which is the case in real-world applications since results generated earlier do not know what results will be generated later).

After the multi-head attention, the original value of the positional encoding is combined with the result of the multi-head attention to allow the positional information to be retained in the data. Then, batch normalization (represented as "normalization" in the diagram) is performed to put the data into a zero-mean, unit-standard-deviation state, which enhances stability in the neural network.

For the "input" channel, the value then passes through a feed-forward fully-connected neural network, and that is added to results before the fully-connected network, and finally normalized. This result represents the encoding from the "input" channel. Such a process can be performed multiple times, and the number of times it is repeated is equal to the number of encoder layers.

The result (of one or more repeats) is received by the "output" channel and joined at a multi-head attention block from the previous data in the "output" channel. Again, the adding of values and batch normalization is performed, then another fully connected neural network, another adding and batch normalization is performed. Such a process starting from the masked multi-head attention to the adding and batch normalization is performed for some repetitions, as represented by the layers of the decoder.

Finally, the result of the "output" channel is computed over a single fully-connected layer of a neural network, and softmax is applied to it. Softmax is an activation function that ensures the sum of all neuron outputs in the softmax layer of the neural network

adds up to one. In other words, this output of one is split up between different neurons based on the size of their input to the softmax function. Since probabilities add up to one, softmax makes an effective activation function when predicting output probabilities, which is the result to be outputted.

2.3.2 Rhythm Prediction. The main innovative point of this research is its ability to effectively guide the acoustic model to generate rhythmic speech. The embedding of pitch and energy is passed to the rhythm predictor, along with the 16 letters prior to the phoneme to predict pausing for better context (if the phoneme doesn't have 16 letters before it, -1 is used as a placeholder), to check whether the previous embedding is suggestive of appropriate pausing in the given context, and if not, such error is back-propagated on the loss function. Through training, the acoustic model will eventually generate audio with appropriate pausing.

This step is done after the pitch and energy prediction because pausing length is highly dependent on what the text is trying to express, which can be modelled to some extent by the pitch and energy predictors. This step is done before the duration prediction because the embedding of duration changes the dimensionality of the embedding vector, which wouldn't match the phoneme-by-phoneme requirement for the pause predictor.

2.3.3 Variance Predictor. The variance predictor receives a latent vector, which is passed through conv1d (one-dimensional convoluted neural network), RPSigmoid, layer normalization [6], dropout layer [7], conv1d, RPSigmoid, layer normalization, and dropout layer in this order and setup is used. Note that RPSigmoid is a new and trainable activation function. RPSigmoid takes the formula of

$$y = \frac{abx}{(1 + b|x|^c)^{\frac{1}{c}}} + dx$$

where a,b,c,d are trainable parameters randomized at the start of the training process.

The usage of convolutional neural networks ensures a low count of parameters while achieving decent quality in synthesized audio. The usage of layer normalization brings the benefits of training stability and faster convergence by ensuring different layers treat data that is approximately within the same magnitude. The usage of a dropout layer prevents the model from overfitting by changing some neurons' output to zero, introducing variation, and forcing the model to learn the patterns despite such perturbations (instead of just memorizing the correspondences of the previous layer's output and the correct answer).

There are two types of math equations: the *numbered display math equation* and the *un-numbered display math equation*. Below are examples of both.

2.4 Vocoder

The acoustic model can directly output the mel spectrogram, but multi-sensory learning requires the use of audio instead of mel spectrogram to stimulate dyslexic learners. To facilitate the transformation from mel spectrograms to audible waveforms, a vocoder is used. This research uses the HiFi-GAN (short for High-Fidelity Generative Adversarial Network) vocoder [8], the current state-of-art approach. HiFi-GAN is a pretrained model, pretrained using one generator rebuilding audio from mel spectrograms and two types of discriminators operating on multi-period and multi-scale bases. Each multi-period discriminators only accepts input from certain discrete time points (multiples of a certain prime number), thus having each discriminator focus on different parts of the audio.There are three individual multi-scale discriminators, each focusing on a different scale: the original data, data that has undergone 2x pooling, and data that has undergone 4x pooling. The usage of multi-scale discriminators introduces the ability to distinguish continuity to the discriminator group since the multi-period discriminators each look at discrete data points.

The HiFi-GAN model is pretrained on LJSpeech, which contains around 24 hours of a single English speaker. HiFi-GAN is built to improve using three losses: GAN (as performed by the aforementioned discriminators), mel-spectrogram (which reconstructs the mel-spectrogram from the produced audio), and feature-matching (a comparison between the generated and ground truth audio samples). The weighting of these losses in the final training process is 1 to 2 to 45 respectively.

3 EXPERIMENTAL SETUP

The experiment to verify the effectiveness of the aforementioned method will be described below. First, the usage of the dataset will be described, which is followed by how the aforementioned method is carried out, and finally a summary of how the quality of the text-to-speech process in this research is evaluated.

3.1 Dataset

This experiment tests the model using the LJSpeech dataset. This dataset consists of around 24 hours of speech performed by one female English speaker. The speech consists of the reading of seven non-fiction books therefore the tone is more informative and less dramatic. The 24 hours of speech is broken up into 13100 audio clips, each within 10 seconds of length. This dataset is around 2.6 GB large.

This dataset is selected because it is one of the most commonly-used datasets in text-to-speech modeling. Moreover, the reasonable size of the dataset (if compared to other datasets such as LibriSpeech) makes it a viable option since the computational capacity is limited.

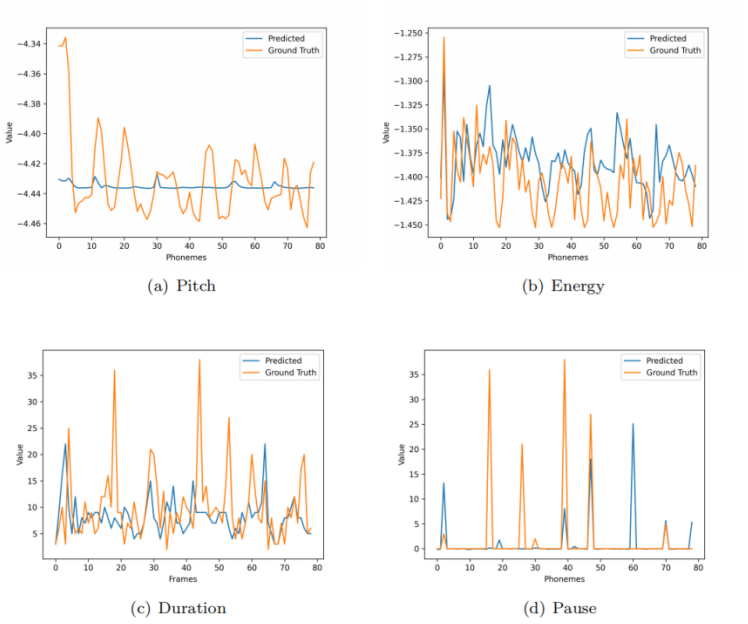
3.2 Training on the Dataset

A NVIDIA Tesla T4 GPU equipped with 15 GB of memory was used for this experiment. The proposed method uses a batch size of 16 along with an epsilon of 10 −9, an optimizer of Adam, betas of 0.9 and 0.8, 2000 steps for warm-up, and trained for a total of 10,000 steps.

3.3 Evaluating the System

For this text-to-speech system, which features high expressiveness in pausing, the following evaluative metrics will be used.

Mean Opinion Scale (MOS): the subjective evaluation of listeners upon the quality of the model- synthesized audio. Twelve listeners (six female and six male) with fluent English capabilities are asked to listen to the generated audio. Each listener is asked to give a score of 1 (the worst, where the audio is unintelligible) to 5 (the audio sounds perfect) to each piece of audio they hear. To ensure fairness,



(a) Pitch (b) Energy

(c) Duration (d) Pause

Figure 7: The comparison between the ground truth and the predicted values for pitch, energy, duration, and pause features on unseen data.

each listener will listen to audio generated from both Fastspeech2 (the baseline) and RhySpeech without knowing the audio-model correspondences (unbiased judgment).

Pause Prediction Effectiveness Statistics: accuracy (the fraction of all entities that are judged correctly), precision (the fraction of those judged positive (paused) that are supposed to be positive), recall (the fraction of those that should be positive (paused) that are judged positive), and F1 values that reflect the effectiveness of the prediction of pausing.

Loss-by-Epoch: this is specifically a comparison to show whether the activation function (its purpose is to aid faster convergence) is effective in helping the model converge.

Model Parameter Count: this is a metric used in reference with the MOS (quality of prediction) to determine whether the acoustic model of this research is deployable, which would require a comparatively small parameter count.

4 RESULTS

After introducing the methods of this research, results will be given to justify the validity of the aforementioned design. The results section will begin by presenting the general outputs of this study. Afterward, changes in synthesized vocal quality after the addition of the pause predictor will be presented. Finally, the usage of the RPSigmoid activation function will be justified.

4.1 General Results

The RhySpeech model was able to achieve a MOS value of 3.87, which means that the audio quality is decent. To offer a context of comparison, the MOS value of the baseline is only 3.73, as shown in Table 2. At first glance, it is already evident of RhySpeech's superior audio quality.

Table 3: The comparison of parameter count (fewer means requires less storage space, therefore is better) between the baseline (Fastspeech2) and RhySpeech.

Model Name	Parameter Count
Fastspeech2 (baseline)	43.8M
RhySpeech	41.4M

To visualize the high quality of generated audio, line graphs demonstrating values of vocal characteristics (pitch, energy, duration, and pause) will be presented for the ground truth and the generated results of RhySpeech in Figure 7. Note that RhySpeech has not trained on these pieces of audio in advance.

The similarity between the ground truth and the generation of RhySpeech shows that RhySpeech is strong at modeling speech characteristics even in unseen scenarios as shown in Figure 7.

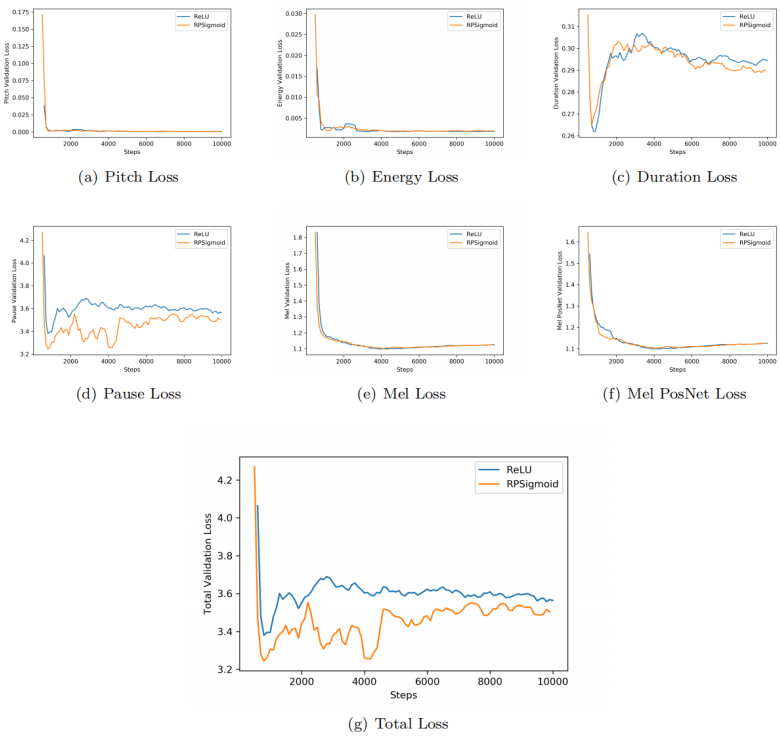
Since the scenario requires a relatively lightweight model for implementation, the parameter count is also presented in Table 3. From the smaller model size of RhySpeech, it is shown that RhySpeech is comparatively small, and therefore more usable in resource-restricted scenarios.

4.2 Pause Predictor Results

To prove that pausing has been effectively captured using the pause predictor, statistical comparisons between the baseline (Fastspeech2) and RhySpeech regarding whether a pause has been predicted will be made in Table 4. The comparison is done based on

Table 4: The statistical quantities that reflect the effectiveness of Fastspeech2 and RhySpeech at predicting the existence of pausing when no punctuation information is available. Note that Fastspeech2 is unable to predict pausing when not given punctuation so no pausing has been predicted, leading to the presence of NaN values. The accuracy for RhySpeech is lower than Fastspeech2 due to the sparseness of pausing (less than 7 percent of the data used to generate the statistics are supposed to be where pausing is present.

Quantity	Baseline (Fastspeech2)	RhySpeech (proposed method)
Accuracy	0.957	0.948
Precision	NaN	0.679
Recall	0	0.430
F1	NaN	0.527



(a) Pitch Loss (b) Energy Loss (c) Duration Loss

(d) Pause Loss (e) Mel Loss (f) Mel PosNet Loss

(g) Total Loss

Figure 8: The loss-per-epoch curves for various parts of the text-to-speech system and the total loss comparing ReLU and RPSigmoid

the LJSpeech validation set (which neither model has been trained on before).

4.3 RPSigmoid Results

To justify RPSigmoid's effectiveness as an activation function in this model, its effects of convergence are compared with the state-of-art activation function in text-to-speech fields: ReLU. In Figure 8.

4.4 Discussion

In conjunction with the successes in this research, there are also potential directions to research into in the future. Firstly, an effort can be made to enhance stability of the quality of generated audio, ensuring that when this text-to-speech system is deployed, people with dyslexia can enjoy a reliable auditory stimulus. Additionally, an approach of taking pausing into the embedding could be attempted, in order to make the pausing prediction directly impact the audio, creating even more natural rhythmic patterns in synthesized speech. Lastly, a user-friendly system can be developed on top of the theoretical basis introduced in this paper, allowing for dyslexic people to access auditory stimulus more conveniently

5 CONCLUSION

This paper proposes a component to model rhythmic information in the human speech by predicting pausing under the Transformer architecture based on pitch and energy embedding. The pausing prediction generates a loss which is used to improve the acoustic

model to produce more accurate pausing patterns. To achieve desirable results under a smaller parameter count, this pause predictor is coupled with RPSigmoid (Randomized and Parameterized Sigmoid) as its activation function. Such a method has resulted in an increase in MOS and pausing recall compared to Fastspeech2, while enjoying the convenience of less parameters than Fastspeech 2.

REFERENCES

[1] Philip Kirby. Dyslexia debated, then and now: A historical perspective on the dyslexia debate. Oxford Review of Education, 46(4):472–486, 2020.
[2] Xu Tan, Tao Qin, Frank Soong, and Tie-Yan Liu. A survey on neural speech synthesis. arXiv preprint arXiv:2106. 15561, 2021.
[3] Michael McAuliffe, Michaela Socolof, Sarah Mihuc, Michael Wagner, and Morgan Sonderegger. Montreal forced aligner: Trainable text-speech alignment using kaldi. In Interspeech, volume 2017, pages 498–502, 2017.
[4] Fabian Pedregosa, Ga¨el Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, *et al.* Scikit- learn: Mfachine learning in python. the Journal of machine Learning research, 12:2825–2830, 2011.
[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, L– ukasz Kaiser, and Illia Polosukhin. Attention is all you need. Advances in neural informationp rocessing systems, 30, 2017.
[6] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. arXiv preprint arXiv:1607. 06450, 2016.
[7] Nitish Srivastava. Improving neural networks with dropout. University of Toronto , 182(566):7, 2013.
[8] Jungil Kong, Jaehyeon Kim, and Jaekyoung Bae. Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis. Advances in Neural Information Processing Systems,3 3:17022– 17033,2020.