Automatic Speech Recognition with Synthetic Speech

Yash Prakash
Department of Mechanical Engineering
Carnegie Mellon University
Pittsburgh, PA 15213
yprakash@andrew.cmu.edu

Thomas Klein
Department of Mechanical Engineering
Carnegie Mellon University
Pittsburgh, PA 15213
tmklein@andrew.cmu.edu

Nikhil.C.G
Department of Mechanical Engineering
Carnegie Mellon University
Pittsburgh, PA 15213
nchinnal@andrew.cmu.edu

Saurav Kambil
Department of Mechanical Engineering
Carnegie Mellon University
Pittsburgh, PA 15213
skambil@andrew.cmu.edu

Abstract

This paper explores a solution to one of the main factors stunting the acceleration of state-of-the-art speech recognition systems. This is the acquisition of high-quality data. To remove excess costs in acquiring the required data, the area of synthetic speech generation and its applications in training is being tested. This paper aims to test the hypothesis of solely synthetic speech being capable of training a speech recognizer that is comparable to a model trained using actual human speech. GitHub: Link

1 Introduction

The largest hindrance to training high-performance automatic speech recognition systems is the collection of well-labeled and curated data, which up to this point has proven to be exceedingly expensive both in time and resources. This project aims to overcome this challenge by determining if automatic speech recognition systems that are solely trained on large amounts of synthetic speech perform at least as well as systems trained on actual human speech. The project will delve deeply into speech synthesizers to identify the best mechanisms to produce a corpus of synthetic human speech that can rival that of actual human speech in speech recognition training.

The paper will explore whether automatic speech recognition systems trained solely on synthetic speech can generalize and operate efficiently in real-world environments. The impact of such a discovery could potentially result in a paradigm shift in speech research, with a greater focus on accurate speech synthesis becoming a priority. Improvements in synthesis could lead to the elimination of costs associated with data collection and curation, allowing for more resources to be directed towards the development and training of superior synthesis and recognition systems.

2 Literature Review

As instances of similar research are sparse the following literature review will comprise some recent State-of-the-art techniques in both speech synthesis and speech recognition as well as circumstances when synthetic speech was used in the training process.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

2.1 Speech Synthesis

Speech Recognition has seen a lot of research done on it, with many different organizations like NVIDIA, Google, and Microsoft. There have been multiple approaches to tackle this problem. Google created a system called Tacotron 2 [13]. This uses the technique of converting the input transcript to mel spectrograms through a series of Convolutional layers and LSTMs. The obtained mel spectograms are then passed through a WaveNet Module to generate the waveform samples. These waveforms achieved a Mean Opinion Score (MOS) of 4.21 which was significantly better than existing Text-To-Speech (TTS) models which have MOS scores of about 1. Google further expanded the field of TTS by releasing the model called AudioLM [2]. This method combines both semantic and acoustic tokens in the audio generation process. This uses the same method as Tacotron 2 to convert the transcripts into semantic tokens, but it also takes in audio tokens that are generated with the help of SoundStream neural codec. These tokens are then converted into a language model that can use these two tokens to generate audio that is trained with adversarial examples and hence is very robust. The latest development on TTS models and the current State-of-the-art is Vall-E [17] by Microsoft. This uses the same transcript as the text sample from the speech generation but also takes in a 3-second audio clip of a speaker which it is trying to mimic. The audio sample is converted into the audio tokens which are then used to replicate the sound pattern of the speaker in the speech from the transcript. This method allows the replication of the voice of the speaker in the speech, thereby accounting for the different accents that are present in the world accurately. This method does not use mel spectrograms like the earlier versions, but instead converts the transcripts and audio sample into neural codec language, which is then converted into audio samples.

2.2 Speech Recognition

Automatic Speech Recognizer(ASR) has been around for a while with a lot of different iterations in the past, with commercial deployment in the case of voice assistants like Siri, Google Assistant, and Alexa. But the benchmark model that is currently being used is wav2vec 2.0 [1] by Facebook AI. This is trained on Librispeech and is composed of transformers and CNNs. The model is trained with labeled data with a Connectionist Temporal Classification (CTC) loss. With a lot of fine-tuning, the model was able to achieve a Word-Error-Rate (WER) of just 1.8% to 3.3%. This is significantly better than the existing competition in the form of just conformers and LSTM-RNN. But the current State-of-the-art for ASR is Open AI's Whisper. This architecture is an implementation of an encoder-decoder Transformer. The input is split into 30-second bits and then converted into mel spectograms. These are then passed into the encoder to give an output which is sent to the decoder. It is trained to give predict the corresponding text captions from the audio.

2.3 Synthetic Speech to Train Speech Recognizer

The state-of-the-art (SOTA) performance on using synthetic speech to train a speech recognizer is constantly evolving as new research is published in this field. However, a few recent studies have achieved impressive results in this area.

For instance, a recent paper by Karita et al. (2019) [6] used a deep learning-based speech synthesizer to generate a large amount of synthetic speech for training a Japanese automatic speech recognition (ASR) system. The authors augmented their dataset with synthesized speech and achieved a 10% relative reduction in word error rate compared to training with only real speech data.

Another study by Zhou et al. (2021) [14] proposed a multi-task learning framework that leverages synthetic speech generated by a Tacotron 2-based speech synthesizer. They achieved state-of-the-art performance on the LibriSpeech dataset and showed that augmenting the training data with synthetic speech can significantly improve ASR performance.

Furthermore, a recent work by Chen et al. (2021) [15] proposed a data augmentation technique that combines synthetic speech with real speech data using a variational autoencoder (VAE). The authors showed that augmenting the training data with synthetic speech generated by their VAE-based system can improve the robustness of the ASR model to noise and speaker variations.

Overall, these studies demonstrate the potential benefits of using synthetic speech to augment training data for speech recognition systems. While the exact SOTA on using synthetic speech to train a

speech recognizer may vary depending on the specific task and dataset, it is clear that synthetic speech can be a valuable resource for improving the accuracy and robustness of ASR systems.

3 Model Description

The previously stated recent developments in both synthesis and recognition have taken unique leaps to expand knowledge within the field of natural language processing (NLP). However, with limited computation power available certain model choice decisions needed to be made. Although these decisions may negatively affect performance, the goal is to show a proof of concept in training the speech recognizer.

3.1 Synthesizer Choice

State-of-the-art synthesis models such as Vall-e [17], Spear-TTS [7], or AudioLM [2] have the potential to produce extremely high-quality synthetic speech data. These models' ability to produce an extremely vast amount of variations in human speech, similar to the variations found in actual speech due to differing accents, may allow for much superior speech recognition performance. Although this is the case, the recency of these impressive text-to-speech models means that no pre-trained models are currently available for producing the large amounts of synthetic speech required. However, the possibility of producing multiple synthetic voices for training is still feasible by using the synthesizer known as VITS [8]. As this enables the input transcripts of text to be spoken in a variety of different pitches and rhythms in an attempt to replicate the many variations in human speech.

3.2 Synthesizer Model

The speech synthesizer known as VITS is an acronym that stands for variational inference with adversarial learning for end-to-end text-to-speech. VITS was developed as a proposed solution to problems found in parallel text-to-speech systems. Two-stage pipelines have been shown to require sequential training or further fine-tuning for high-quality synthetic speech. This speech synthesizer looks to apply a combination of many methods to achieve high-quality speech waveforms and optimal training. A variational autoencoder (VAE) [8] is first used to allow more efficient learning of the model. The model then applies normalizing flows to the conditional prior distribution as well as the adversarial training. Finally, the model utilizes a proposed stochastic duration predictor which is used to capture the variations in speech that are vital in the creation of all generalizable speech recognition systems [8]. These major components within the VITS pipeline mentioned above will be explored further below.

3.3 Posterior Encoder

The posterior encoder utilizes the non-causal WaveNet residual blocks [11], as these blocks are also found in WaveGlow [12] and Glow-TTS [9]. These particular residual blocks consist of layers of dilated convolutions that expand the kernel by inserting holes between elements in the kernel. This is then followed by a gated activation unit, which is used to control what information will be passed to the next layer, as well as the skip connection found in residual networks. After the residual blocks, there is a linear projection layer that is used to produce the mean and variance of the normal posterior distribution. In the case of multi-speakers, global conditioning, found in [11], is also used within the residual blocks to add speaker embeddings.

3.4 Prior Encoder

The prior encoder consists of a text encoder that processes the input phonemes as well as the normalizing flow. The text encoder is the same encoder found in a transformer [16] and utilizes relative positional encoding. This encoder obtains the hidden representation of the input through the text encoder and projection layers. The normalizing flow layer is a stack of affine coupling layers consisting of the same WaveNet residual blocks mentioned above in the posterior encoder. Also similar to the previous encoder in the case of multiple speakers, a speaker embedding is added to the residual blocks through the use of global conditioning.

3.5 Decoder

The decoder being used is nearly identical to the HiFi-GAN V1 generator proposed in [10]. The decoder network consists of a stack of transposed convolutional layers, each of which is followed by a multi-receptive field fusion module (MRF). The MRF returns the sum of the outputs from multiple residual blocks representing varying receptive field sizes. The decoder also utilizes an additional linear layer to transform speaker embeddings to be later added to the input latent variables for the multi-speaker setting.

3.6 Discriminator

The discriminator being used is a replica of the multi-period discriminator proposed in [10] alongside the HiFi-GAN model. A multi-period discriminator is a mixture of sub-discriminators that each accept equally spaced sample intervals of the input audio. The sub-discriminators attempt to capture different implicit structures by looking at the different intervals within the input audio.

3.7 Stochastic Duration Predictor

The stochastic duration predictor is tasked with estimating the distribution of phoneme duration from the conditional input. The predictor consists of a stack of residual blocks with both dilated and depth-separable convolutional layers as well as neural spline flows [5]. Neural spline flows perform better at improving the transformation expressiveness than affine coupling layers with a comparable number of parameters. Similar to the previous components within the model, the stochastic duration predictor also utilizes a linear layer that transforms the speaker embeddings and adds them to the input.

3.7.1 Visualization

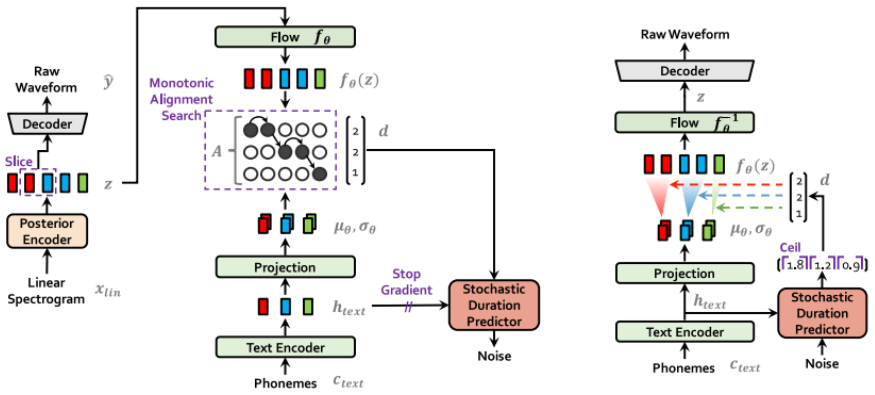

Figure 1: VITS Architecture [ViTS citation]

Figure 1 provides an outline of the previously described VITS speech synthesizer. The training procedure is indicated by (a), whereas the inference procedure is denoted by (b). This model can be seen as a conditional variational autoencoder, which consists of a posterior encoder, decoder, and conditional prior with a stochastic duration predictor. The conditional prior in this case is comprised of the green blocks, which are the normalizing flow, linear projection layer, and text encoder.

3.8 Recognizer Choice

As previously determined in the literature review there is a wide range of state-of-the-art models to choose from as speech recognition has been one of the front runners in natural language processing (NLP) research. For the purposes of this project, it was believed that the speech recognition model, Listen, Attend and Spell [3], or LAS for short, would be a suitable choice as the model has been

well-documented and well-tested on the corpus intended to be used, LibriSpeech [**librispeech**]. In conjunction with this, the model size is also a major consideration in model choice. With limited computation available LAS provides satisfactory results without the large computing requirements to train state-of-the-art models. This decision seemed appropriate as the goal is to test the hypothesis that synthetic data is capable of achieving similar results to that of ASR systems trained on real data.

3.9 Recognizer Model

The Listen, Attend and Spell architecture follows the popular architecture found in NLP consisting of an encoder and decoder system. More specifically the "Listener" encodes the inputted audio frames, MFCCs in this case, and encodes them into a higher-level representation. With a similar naming convention, the "Speller" is decoding the encoder output into probability distributions of each of the characters at each step. The decoder also has the component of an attention mechanism where the final "Attend" component lies. This component conditions the probability distribution of the next characters in the sequence on the previously seen characters.

3.9.1 Visualization

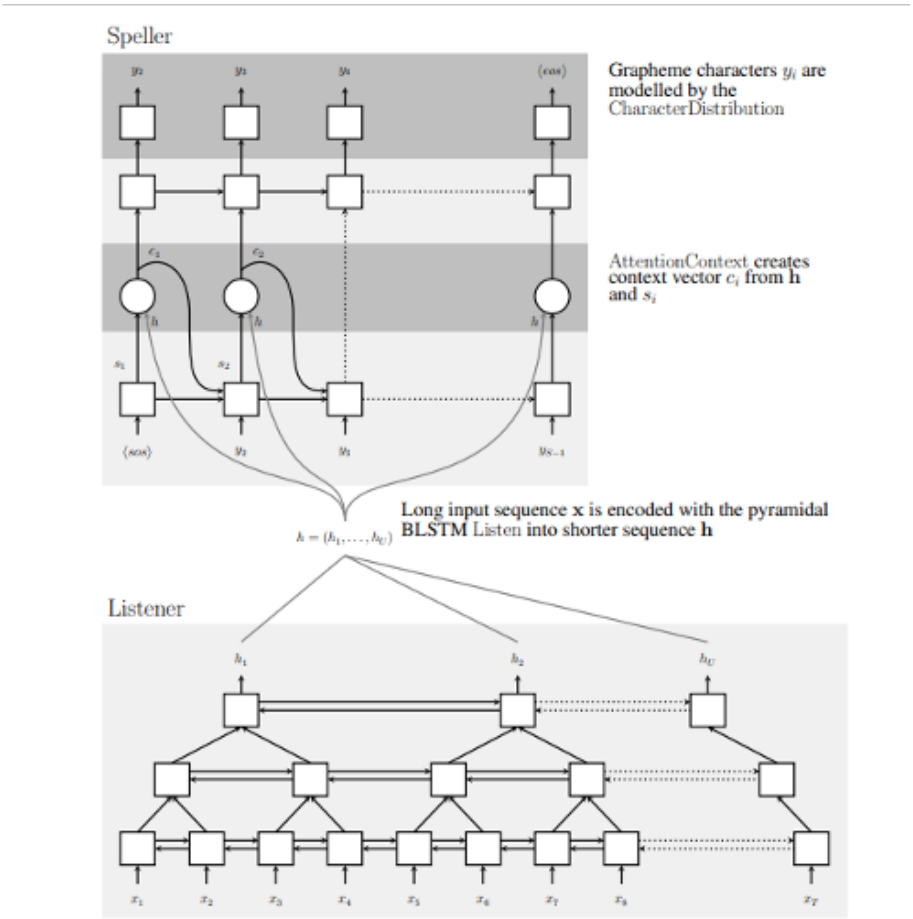

Figure 2: LAS Architecture [LAS citation]

Figure 2 outlines the Listen, Attend, and Spell architecture mentioned previously. The listener, located at the bottom of the image, consists of 3 pyramidal BiLSTM layers that encode the input into higher-level features. The upper portion is the speller and decodes the latent representations by using the previously mentioned attention-based LSTM transducer.

3.9.2 Listen

As previously mentioned the Listen operation within the encoder is tasked with obtaining a latent representation of the audio. The overall form of the encoder utilizes Pyramidal Bidirectional Long Short Term Memory (pBLSTM). The pBLSTM is a variant of the original Bi-LSTMs that downsamples the time resolution by a factor of 2 through the process of concatenating adjacent pairs of inputs. These newly concatenated pairs are then fed through the conventional Bi-LSTM. The goal of this is to significantly reduce the time resolution such that the attention module can extract the relevant features in fewer time steps. This also has the effect of reducing the computational complexity thus increasing the efficiency of both model learning and inference.

3.9.3 Attend and Spell

As previously mentioned the Attend and Spell decoder utilizes an attention-based LSTM transducer [4] that, predicts the probability distribution of each character given the previously seen characters. The attention module generates a context vector corresponding to each time step. These context vectors are formed when the previous decoder states are linearly blended with the outputs of the encoder. The "attended" context vector is then used to produce the character probability distribution through the use of a multi-layer perceptron with a softmax output.

4 Experimental Evaluation

4.1 Dataset

The dataset chosen for the purposes of training and testing the LAS Model, as well as the synthetic data generation was the LibriSpeech dataset [**librispeech**]. This dataset is a corpus of entirely English speech that was formed and curated for speech recognition. The data itself was derived from the LibriVox Project, consisting of roughly 1000 hours of speech sampled at 16 kHz.

4.1.1 Generation

As the goal is to train the LAS model using synthetic data. The first step in the process is creating an identical corpus of LibriSpeech of entirely synthetic speech. This is accomplished by taking the transcripts from the train-clean-100, train-clean-360, dev-clean, and test-clean subsets and passing them through the pre-trained VITS synthesizer. This large amount of synthetic data allows for parsing of different sizes to be used during training.

4.1.2 Preprocessing

For training to be effectively underway after the production of the synthesis data, a few preprocessing steps were first required, the first steps were to format the transcripts so that the true labels may be more effective than the predicted sequences. This formatting consisted of converting the initial string to a list of characters as the output of the LAS decodes the most likely character at each time step, producing the predicted sequence. It also consisted of converting the .txt transcript files to .npy files to be fed into the recognizer. The subsequent step in preprocessing transcripts is to add the Start-of-Sequence (SOS) and End-of-Sequence (EOS) tags to the converted npy files. For the input of the LAS model being used, the audio files were converted into the corresponding MFCCs, also known as Mel-frequency cepstral coefficients. This provides an effective method to extract features from the waveform. These MFCCs consisted of 27 features and were extracted using the librosa python package, a popular music and audio analysis library. The sampling rate of the audio files also needed to be modified to match the 16kHz value required by the LAS mode. Once these steps were complete, the training was then ready to commence.

4.2 Implementation

Our team utilizes synthetic data created by a speech synthesizer to train our speech recognition model. To measure the general trend of loss and accuracy by adding more data, we split our data into five different groups. We would be training five models: Real Libri-Speech train clean 100 data, Synthetic Libri-Speech train clean 300, Synthetic Libri-Speech train clean 200, Synthetic Libri-Speech train

clean 100, Synthetic Libri-Speech train clean 65. Partitioning and using various models not only gives us valuable insight into the increase in accuracy vs the increase in data, but would also provide information about the differences in convergence time, and help weed out any outliers in the model as well.

Further, to make our automatic speech recognition (ASR) model more robust, we've decided to augment our audio dataset with data and spectral augmentation. This is achieved with the addition of frequency and time masking on the audio data before it is fed into the recognizer model. Adding to it, the VITS generator also adds noise to the synthetic data. The generation of synthetic audio changes this noise variable on the generator function to utilize a variety of noises. This also prevents the recognizer model to have a bias towards particular noise.

The LAS recognizer function is also trained on the Libri-Speech Train Clean 100 audio to allow us to compare our results with the performance of the real data. This would help us provide a better benchmark. We are training this data on the same Recognizer architecture under the same parameters for the same number of epochs. This ensures the normalization of all external factors enabling us to get unbiased results.

The LAS Model uses the Cross-Entropy loss function to define the training loss for the data per epoch. The cross-entropy loss function for a single training example with actual label y and predicted probability distribution p is defined as:

$$\mathcal{L}(y, p) = -\sum_{i} y_i \log(p_i) \tag{1}$$

where i indexes the classes, y_i is the actual probability of the ith class, and p_i is the predicted probability of the ith class.

The overall cross entropy loss for a dataset of n training examples is the average of the individual cross entropy losses:

$$\mathcal{L} = \frac{1}{n} \sum_{j=1}^{n} \mathcal{L}(y_j, p_j) \tag{2}$$

where y_j and p_j are the actual and predicted probability distributions for the jth training example.

The cross entropy loss function penalizes the model more heavily for predictions that are far from the actual label. The logarithm in the formula ensures that the penalty grows exponentially as the predicted probability deviates further from the actual label.

The LAS Model also uses Levenshtein Distance while calculating the loss for the Validation dataset. Levenshtein distance is distance between two strings a and b, denoted as $lev_{a,b}(i, j)$, represents the minimum number of insertions, deletions, or replacements required to transform the substring $a[1 : i]$ into the substring $b[1 : j]$.

The base case is when either i or j is 0. If one of the substrings is empty, the distance is the length of the other substring, since all characters in the non-empty substring need to be either inserted or deleted.

Otherwise, we consider the three possible operations that can be performed to the substring $a[1 : i]$ to transform it into $b[1 : j]$:

- Insertion: We insert the character b_j into a, so that $a[1 : i]$ becomes $a[1 : i - 1] + b_j$. This increases the distance by 1.
- Deletion: We delete the character a_i from a, so that $a[1 : i]$ becomes $a[1 : i - 1]$. This increases the distance by 1.
- Replacement: We replace the character a_i with b_j, so that $a[1 : i]$ becomes $a[1 : i - 1] + b_j$. This increases the distance by 1 if a_i and b_j are different, and 0 otherwise.

We choose the operation that results in the minimum distance and update the current $lev_{a,b}(i, j)$ value accordingly. We do this recursively for all values of i and j until we obtain the distance between the entire strings a and b, which is given by

$$lev_{a,b}(len(a), len(b)). \quad \text{lev}_{a,b}(i,j) = \{\max(i,j) if \min(i,j) = 0,$$
$$op \in \{[a_i \to b_j], , \} min \begin{Bmatrix} lev_{a,b}(i-1,j)+1, \\ lev_{a,b}(i,j-1)+1, \\ lev_{a,b}(i-1,j-1)+[a_i \neq b_j] \end{Bmatrix} otherwise.$$

Further, while testing our model with Real Speech from Libri-Speech Test Clean, we used the word error rate metric to determine the accuracy of our model against real speech. The Word Error Rate (WER) is a metric used to evaluate the performance of automatic speech recognition systems. It measures the percentage of incorrect words in the output of the system compared to the reference transcript.

Let S be the total number of words in the reference transcript, D be the total number of words in the output of the system, and I be the total number of insertions, deletions, and substitutions required to convert the output into the reference transcript.

Then the WER can be calculated using the following equation:

$$WER = \frac{I+D+S}{S} \times 1/100$$

Alternatively, we can express the WER as the sum of the substitution, deletion, and insertion rates:

$$WER = \frac{S_{sub}+S_{del}+S_{ins}}{S} \times 1/100$$

where S_{sub} is the number of substitutions, S_{del} is the number of deletions, and S_{ins} is the number of insertions.

5 Results

As previously mentioned, four separate models were trained and tested on the original test-clean dataset from LibriSpeech. These four models each had separate training sets, these being the original train-clean-100, as well as three separate partitions of the newly created synthetic data. These four partitions contain 65, 100, 200, and 300 hours respectively. In using WandB, a Python package utilized in tracking model training, the validation curves can be visualized, and a relative idea of convergence. This visualization can be seen below. An important note in this visualization is that although validation began converging similarly to that of the model with the real data, the validation set used in conjunction with the synthetically trained models also consists of synthetic data.

5.1 Validation Results

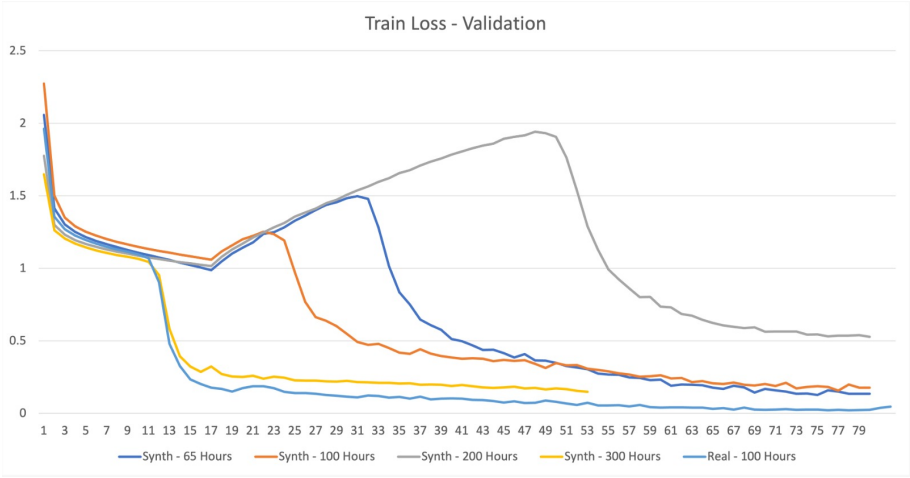

Figure 3: Training Loss: Cross Entropy

As seen in the various visuals, the models generalized very well on their respective datasets. The synthetic data models achieved a good score on the synthetic validation data, and so did the real

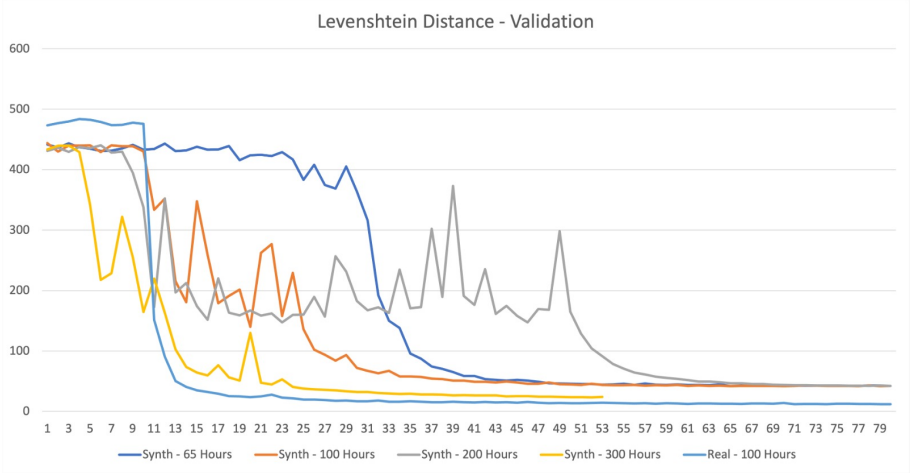

Figure 4: Validation Distance: Levenshtein

Table 1: Levenshtein Distance and Training Loss on dev-clean

Models	LD	Loss
LAS-100-Real	12.11	0.02
LAS-100-Synth	41.80	0.15
LAS-200-Synth	42.16	0.52
LAS-300-Synth	23.38	0.14

speech data model. It should also be noticed, that with the increase of data, the loss and Levenshtein Distance reduced. Further, the convergence time also seemed to drop as the amount of data increased.

5.2 Test Results

Although the above figure represented results on separate validation sets, the next step was to identify the actual performance of the synthetically trained models on real-world data as found in the test-clean dataset. In doing so the following table below shows each model's performance in the two desired metrics of Levenshtein distance and WER, when testing on real human speech.

Table 2: Levenshtein Distance and WER on test-clean

Models	LD	WER
LAS-100-Real	13.08	18.24
LAS-100-Synth	483	99.5
LAS-200-Synth	373	77.5
LAS-300-Synth	473	99.7

6 Conclusion

As concluded from the initial experiments in synthetic training the overall results leave much to be desired to compete with real-world data and the performance it gives. This is likely due to the intricacies found within speech that are being replicated within the speech synthesizer. As this is the case it is suspected that further pre-processing to either the original audio waveforms or after they are converted into MFCCs. Since the amount of synthetic data shows a negative trend in performance with hours of synthetic data it can be assumed that the quality of synthetic data in comparison to the original real data is vastly different. This can be identified due to the fact that as the number of training hours increases the corresponding Levenshtein distance and word error rate. This vast difference may not be distinguishable for the human ear but for training a speech recognizer this

results in extremely poor performance. Although this trend exists, further testing may prove beneficial in adding a significant amount more noise, outside the noise claimed to be added within the VITS architecture, to the synthetic data to enable the models to experience improved generalization.

There are various types of potential noise, including background noise, distortion, reverberation, and channel noise. Adding background noise allows the ASR model to handle real-world scenarios with background noise present. Distortion introduces modifications to the audio signal, allowing the model to handle different levels of clarity and quality. Conversely, reverberation helps the model handle speech recorded in different environments like small rooms or large halls. Finally, simulating various types of channel noise, such as packet loss, jitter, or delay, enables our model to handle other communication channels better.

In conjunction with adding a significant amount more noise to the newly generated datasets, the newly trained models also raise a new question for future experiments. That is whether or not these existing synthetically trained models may be sufficient as pre-trained models for future fine-tuning with smaller amounts of real data. Although this would not solve the issue outright, which was the foundational reason for these experiments, it is a step in the right direction and can potentially reduce the costs associated with obtaining and curating large amounts of data. These results also inform us about the inability of the recognizer function to generalize the synthetic data over real data. Though this can be reduced by adding noise making the model more robust, an alternative approach is to use SOTA synthesizers such as Vall-E.

7 Proposed Extensions

In the upcoming phase of our automated speech recognition system development, we intend to employ a pre-trained speech synthesizer with the Vall-E architecture. This synthesizer will enable us to generate a vast amount of synthetic speech that can be used to augment our existing training data, improving our system's accuracy and robustness. To increase the diversity of the generated data, we plan to utilize several augmentation techniques, including pitch shifting, adding noise, perturbing speech speed, time stretching. This approach will introduce variability into the training data, which is essential to ensure that the system is capable of recognizing different speaking styles and accents.

We aim to achieve state-of-the-art accuracy levels by training Whisper from OpenAI from scratch, leveraging the augmented dataset generated by our speech synthesizer. By using this approach, we aim to address some of the challenges that arise in speech recognition, such as background noise, speech rate variations, and speaker variability.

In summary, our plan is to incorporate a powerful speech synthesizer with advanced augmentation techniques to increase the diversity and quantity of training data. This approach will help us to achieve our ultimate goal of developing a highly accurate and robust automated speech recognition system.

References

[1] Alexei Baevski et al. "wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations". In: *CoRR* abs/2006.11477 (2020). arXiv: 2006.11477. URL: https://arxiv.org/abs/2006.11477.

[2] Zalán Borsos et al. *AudioLM: a Language Modeling Approach to Audio Generation*. 2022. arXiv: 2209.03143 [cs.SD].

[3] William Chan et al. "Listen, Attend and Spell". In: *CoRR* abs/1508.01211 (2015). arXiv: 1508.01211. URL: http://arxiv.org/abs/1508.01211.

[4] Jan Chorowski et al. "Attention-Based Models for Speech Recognition". In: *CoRR* abs/1506.07503 (2015). arXiv: 1506.07503. URL: http://arxiv.org/abs/1506.07503.

[5] Conor Durkan et al. *Neural Spline Flows*. 2019. arXiv: 1906.04032 [stat.ML].

[6] Shigeki Karita et al. "A Comparative Study on Transformer vs RNN in Speech Applications". In: *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. 2019, pp. 449–456. DOI: 10.1109/ASRU46091.2019.9003750.

[7] Eugene Kharitonov et al. *Speak, Read and Prompt: High-Fidelity Text-to-Speech with Minimal Supervision*. 2023. arXiv: 2302.03540 [cs.SD].

[8] Jaehyeon Kim, Jungil Kong, and Juhee Son. "Conditional Variational Autoencoder with Adversarial Learning for End-to-End Text-to-Speech". In: *CoRR* abs/2106.06103 (2021). arXiv: 2106.06103. URL: https://arxiv.org/abs/2106.06103.

[9] Jaehyeon Kim et al. *Glow-TTS: A Generative Flow for Text-to-Speech via Monotonic Alignment Search*. 2020. arXiv: 2005.11129 [eess.AS].

[10] Jungil Kong, Jaehyeon Kim, and Jaekyoung Bae. "HiFi-GAN: Generative Adversarial Networks for Efficient and High Fidelity Speech Synthesis". In: *CoRR* abs/2010.05646 (2020). arXiv: 2010.05646. URL: https://arxiv.org/abs/2010.05646.

[11] Aäron van den Oord et al. "WaveNet: A Generative Model for Raw Audio". In: *CoRR* abs/1609.03499 (2016). arXiv: 1609.03499. URL: http://arxiv.org/abs/1609.03499.

[12] Ryan Prenger, Rafael Valle, and Bryan Catanzaro. "WaveGlow: A Flow-based Generative Network for Speech Synthesis". In: *CoRR* abs/1811.00002 (2018). arXiv: 1811.00002. URL: http://arxiv.org/abs/1811.00002.

[13] Jonathan Shen et al. "Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions". In: *CoRR* abs/1712.05884 (2017). arXiv: 1712.05884. URL: http://arxiv.org/abs/1712.05884.

[14] László Tóth et al. "Multi-Task Learning of Speech Recognition and Speech Synthesis Parameters for Ultrasound-based Silent Speech Interfaces". In: Sept. 2018. DOI: 10.21437/Interspeech.2018-1078.

[15] Zhongwen Tu et al. "A Feature Fusion Model with Data Augmentation for Speech Emotion Recognition". In: *Applied Sciences* 13 (Mar. 2023), p. 4124. DOI: 10.3390/app13074124.

[16] Ashish Vaswani et al. "Attention Is All You Need". In: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.

[17] Chengyi Wang et al. *Neural Codec Language Models are Zero-Shot Text to Speech Synthesizers*. 2023. arXiv: 2301.02111 [cs.CL].