TTS Models: Overview, Architecture, and Technical/Ethical Concerns

Judah Nouriyelian (Judahn@seas.upenn.edu)
Project Advisor**:** Professor Dan Roth (danroth@seas.upenn.edu)

Senior Capstone Thesis Draft (CIS 498)
University of Pennsylvania School of Engineering and Applied Science
Department of Computer and Information Science 05/06/24

Abstract

Within the past few years, rapid progress has been made both in the field of NLP (Natural Language Processing) and in the development of new machine learning models. We provide an in-depth analysis of Text-to-Speech (TTS) models, focusing on their architecture, technical intricacies, and the associated ethical considerations. We trace the evolution from traditional concatenative synthesis to contemporary deep learning models like Generative Adversarial Networks (GANs) and diffusion models, examining their unique architecture, the underlying principles of operation, and the improvements they offer over traditional TTS methods. We examine and discuss three-stage, two-stage, and end-to-end frameworks in TTS systems, emphasizing the various means through which TTS systems operate and considerations for each framework. We also provide an overview on the ethical considerations of such technology such as the rise of audio deepfakes as well as the associated technical challenges these new models motivate including data diversity, explainability, and real-time applications. We also explore the vast commercial potential of TTS models in various industries, from audiobooks to accessibility tools. Our work aims to provide insight into the current state and future trends of TTS technologies, emphasizing their potential and the challenges ahead. We aim to provide a comprehensive understanding of modern TTS models, highlighting their potential impact on technology and society.

Introduction

Recent years have seen a surge in the power and applicability of AI (artificial intelligence) models. Deep learning models, those which extract higher level features from the input using multiple layers, have been applied to various different fields with strong success such as advent of the transformer[1], allowing for high accuracy models for protein folding prediction and chatbots[2][3], and diffusion models allowing for text-to-image generation[4]. Moreover, the field of Natural Language Processing, the ability for computers to understand and interpret human language, has undergone a renaissance, with text-to-speech, machine translation, and speech recognition tasks finding themselves at the forefront of both industry and academia[5].

Text-To-Speech specifically has undergone a drastic change in the methodologies of the models. While work in concatenative synthesis[6], which combines short samples of audio to form speech, and statistical parametric synthesis[6], which models aspects of speech production and then generates speech waveform from these models, have become traditional approaches to the task, modern models utilizing diffusion and Generative Adversarial Networks (GANs) have entered both the literature and commercial applications[7]. Diffusion models have gained attention for their ability to generate high-quality speech. These models are based on the concept of iteratively refining a noise signal to produce speech waveforms. By modeling the conditional probability distribution of the signal at each step, diffusion models can generate highly natural-sounding speech. They have been used in applications where naturalness and expressiveness are paramount[8]. GANs, which have seen success in various domains, have also made their way into TTS. GANs can be used to generate speech waveforms by training a generator network to produce speech samples and a discriminator network to distinguish between real and generated speech. GAN-based TTS models often produce high-fidelity and expressive speech with improved realism[9].

Overview of TTS

Audio Background

Sound, a mechanical wave phenomenon characterized by particle oscillations within mediums such as air or water, is digitally represented as a sequence of discrete numerical values, capturing its amplitude and frequency characteristics for electronic processing, storage, and playback[10]. This digital representation enables sound to be processed, stored, and played back electronically. The process employed to convert sound from its analog form into a digital format is known as pulse code modulation (PCM).

In PCM, the analog audio signal is sampled at uniform intervals. This means that the amplitude of the audio signal is measured and recorded at consistent times, producing a series of discrete samples. Each of these samples corresponds to the amplitude of the sound wave at a specific point in time. The frequency at which these samples are taken is referred to as the sampling rate. It's a crucial aspect of digital audio since it directly influences the quality and accuracy of the digital representation.

The Nyquist theorem states that in order to accurately reproduce an analog signal in a digital format, the sampling rate must be at least twice the highest frequency present in the original analog signal. This is essential to avoid aliasing, a form of distortion that occurs when high-frequency components of the analog signal are inaccurately represented in the digital domain. By adhering to the Nyquist criterion, the integrity of the original sound can be maintained in its digital form, ensuring that the reproduced sound closely matches the original analog signal. This digital representation of the waveform encoded using PCM is referred to as “raw audio.” Processing raw audio requires substantial computational resources, which can be inefficient and impractical for many applications, especially those requiring real-time analysis or running on resource-constrained devices. Moreover, the raw audio does not directly correspond to the perceptual characteristics that the human ear and brain are most sensitive to. Our auditory system does not perceive all frequencies equally; we are more sensitive to changes in certain frequency bands than to changes in others. This discrepancy means that processing raw audio may not be the most effective way to analyze sounds from a human auditory perspective, which is especially relevant in audio and speech synthesis applications.

Mel Spectrogram

Mel Spectrograms serve as a critical feature in audio signal processing, effectively capturing the unique characteristics of sound by mirroring the human auditory system's response, thus facilitating improved performance in speech synthesis tasks[11][12]. This is performed by first going through the process of pre-emphasis, transforming the signal by increasing the energy

(amplitude) of the high frequency parts of the signal. This is done by passing the signal to a filter as following

$$y[n]=x[n]-\alpha x[n-1]$$

where y[n] is the output sample, x[n] is the original sample, x[n-1] is the previous sample, and α is a coefficient (typically between 0.95 and 0.97). By subtracting a fraction of the prior sample, for lower frequencies, where there are subtle changes between consecutive samples, y[n] is diminished, while for higher frequencies, where there are rapid changes between consecutive samples, y[n] is amplified. This is done as in natural speech, higher frequencies tend to have lower energy compared to lower frequencies. This imbalance can make it difficult for the subsequent stages of the process, so this correction helps balance the clarity of the speech.

Next, in framing, the continuous audio waveform is divided into short, overlapping segments or "frames." This is necessary because audio signals change over time, and by analyzing short segments, we can assume that the audio characteristics are relatively stable within each segment. Typically, each frame is about 20 to 40 milliseconds long. Overlapping is used to ensure continuity between frames and avoid missing any part of the signal. After framing, we then apply a window function to manage the discontinuities in each frame as these discontinuities at the ends of each frame are artificial and unnatural, causing spectral leakage (the leakage of the signal well beyond its start and end point) later on in the process. The window function, often a Hamming or Hann window, smoothly tapers the ends of each frame to zero, thereby minimizing these discontinuities. The Hamming window is given by

$$w[n] = 0.54 - 0.46 * \cos((2*pi*n) / (N - 1)), \text{ for } 0 \leq n \leq N-1$$

which when multiplied by the original signal gives us the new signal.

The Fast Fourier Transform (FFT) is then applied to each new frame in order to translate the signal from its raw audio waveform to the frequencies within the signal. The discrete Fourier transform (DFT) of a signal are the raw frequencies that are present within the audio form and the FFT is simply an algorithm to calculate the DFT of a signal quickly (faster than the $O(n^2)$ time by application of the definition of the DFT)[13]. The DFT is given by

$$X[k] = \Sigma \text{ (from } n=0 \text{ to } N-1) \text{ of } x[n] * e^{(-i2\pi nk/N)}$$

Where X[k] represents the DFT output at index k, x[n] is the nth input sample, and N is the total number of samples.

After the application of the DFT to each windowed frame of the signal, we obtain a sequence of complex numbers representing the signal in the frequency domain. The next crucial step in audio signal processing, particularly in the context of extracting spectrograms, involves computing the power spectrum of each frame. The power spectrum provides a measure of the power or energy present in each frequency component of the frame, which is essential for analyzing the content of the signal. The power spectrum P[k] of a frame is calculated from the DFT output X[k] as follows:

$$P[k] = |X[k]|^2$$

Now the individual frequencies are mapped to the mel-scale, which is the scale of pitches that humans roughly judge to be equidistant, thus normalizing the audio to a more human perception of the audio[14]. This scale corresponds to a linear spacing of frequencies below 1000Hz and a logarithmic spacing of frequencies above 1000Hz. As a reference point, we define the pitch of a frequency of 1000Hz to be 1000 mels. As such, we utilize the following formula to approximate the mels for a given frequency f in Hz

$$Mel(f) = 2595*\log_{10}(1 + f/700)$$

with the f/700 causing approximate linear growth until 700Hz with logarithmic growth at higher frequencies and the 2595 serving to normalize the scale to the reference point. We create a filter bank with triangular filters equally spaced (for different amounts of Mels) and use this filter bank to extract the energy for each frequency, giving us the frequency content in the mel scale[15]. This thus gives us the way each frequency range changes over time[16]. This ultimately gives us the mel-spectrogram of the audio signal for later use. Figure 1 demonstrates this relationship between frequency and the corresponding Mels, emphasizing the logarithmic relationship in how humans perceive sound.

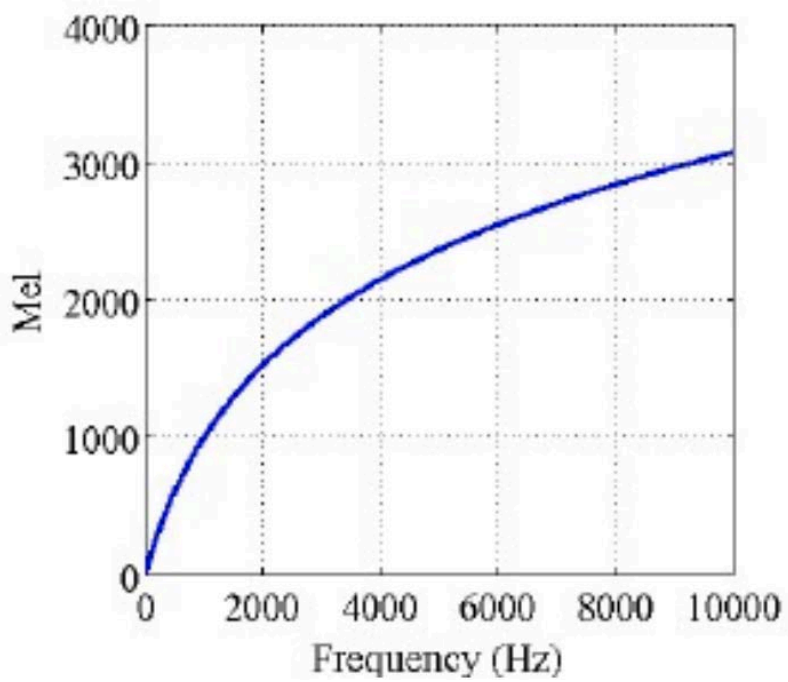

Figure 1. Mapping between frequency and Mels [11]

Traditional TTS Systems

Concatenative and formant synthesis represent the core methodologies in traditional TTS systems, each with distinct approaches to simulating human speech. Concatenative synthesis, a

key method in traditional Text-to-Speech (TTS) systems, involves piecing together small pre-recorded segments of human speech, known as units, to form words and sentences. These segments can range from phonemes, the smallest speech sound units, to entire words or phrases. The primary advantage of concatenative synthesis is its ability to produce highly natural-sounding speech, as the units are sourced directly from human recordings[17]. As such, signal processing and various algorithms are used in order to smooth the waveform in between the concatenated speech units. Hence, the quality of the generated speech is higher when using larger units, though this allows for less flexibility in the overarching TTS system.

Concatenative speech synthesis is classified into three main types: diphone-based, corpus-based, and hybrid synthesis systems. Diphone-based synthesis uses pairs of adjacent phonemes called diphones, representing all possible combinations of phonemes and relies on signal processing techniques like Pitch Synchronous Overlap-Add (PSOLA) and its variants to adjust prosody and naturalness. In PSOLA, the speech signal is broken down into smaller windows to extract short signals (such as the use of the FFT) and pitch correction is done by moving these signals closer together or further apart to modify the pitch of each segment. Moreover, prosody is matched through the duplication and removal of signals to match the duration required in the output speech. Ultimately, these signals are added together utilizing overlapped addition of the signals. Corpus-based synthesis, or unit selection synthesis, stores a large database of speech units and employs exhaustive searches to find the optimal units that minimize target and concatenation costs, producing highly natural speech at the expense of storage requirements. Target costs are computed through a linear combination of the similarity of individual features (e.g. pitch, energy, etc) of the target unit and the units within the corpus[17].

Hybrid synthesis combines statistical and concatenative methods, interweaving statistically generated speech segments to smooth transitions between natural speech units, thereby leveraging the strengths of both approaches while managing potential quality issues due to limited inventory. The statistical method employed is most often Hidden Markov Model (HMM) synthesis which involves training statistical models on a corpus of recorded speech data. The training phase extracts acoustic features (e.g., cepstral coefficients, pitch, and duration) from the speech corpus and uses them to estimate the parameters of HMMs. These models then serve as the basis for synthesizing speech. In the synthesis phase, the target utterance is transformed into a sequence of HMMs that predict the appropriate sequence of acoustic features for the desired speech. The synthesized speech is generated using these predicted features, resulting in a more flexible and adaptable speech synthesis approach[17].

Formant synthesis is one of the earliest methods of speech synthesis, often referred to as synthesis by rule. The technique models the vocal tract transfer function by simulating formant frequencies and amplitudes. Formants are resonant frequencies in the vocal tract that change as the shape and configuration of the vocal tract changes. These changes in formant frequencies

help define different speech sounds. The core concept of formant synthesis involves a source-filter model that uses mathematical representations of the human speech apparatus. In this model, the sound is generated from a source: periodic for voiced sounds (like vowels) and random noise for unvoiced sounds (like fricatives). The generated sound then passes through a model of the vocal tract, which simulates the impact of the oral and nasal cavities on the speech waveform. This technique's strength lies in its simplicity and the small computational footprint required for its execution, making it suitable for embedded and mobile applications[18].

Despite their groundbreaking role, traditional TTS systems face significant limitations in naturalness, expressiveness, and language versatility, highlighting the need for technological advancement. In particular, formant synthesis produces highly intelligible speech, even at high speeds, though formant synthesis often produces speech that sounds robotic or artificial due to the difficulty of accurately modeling natural variations in human speech. HMM synthesis, despite requiring low memory use and high CPU load, often struggles with naturalness due oversimplified vocoders failing to properly model the vocal tract. Moreover, the statistical methods used often lead to over-smoothing of acoustic parameters, which contributes to a loss of fine detail in the synthesized speech. This oversmoothing results from averaging the acoustic features during training, leading to smoothed-out transitions between speech sounds that lack the natural variability found in human speech. In contrast, diphone synthesis, while generally more natural, frequently suffers from discontinuities at the boundaries between diphones, especially at the interfaces between vowel sounds. This can produce perceptible bi-vocalic effects that affect the naturalness and quality of synthesized speech. In unit-selection synthesis, naturalness is often degraded due to the inability to easily vary the speech characteristics, such as duration, pitch, and intensity[18][17][19].

Introduction to Deep Learning in TTS

Neural networks consist of layered architectures of interconnected nodes, which systematically process and analyze data to recognize patterns and perform complex computational tasks without explicit programming. Neural networks, with their multi-layered structures of interconnected nodes, are designed to process and analyze data, recognizing patterns and completing complex tasks without being explicitly programmed. The progression of deep learning within TTS technologies has moved from initial explorations with neural networks and hybrid models to significant advancements in deep learning architectures. This evolution led to the creation of end-to-end models, markedly improving the quality and naturalness of synthesized speech, representing a significant leap in how machines generate human-like speech[8].

Deep Learning Models in TTS

Machine Learning

Machine learning (ML) is a subset of artificial intelligence that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. It focuses on the development of algorithms that can access and use data to identify patterns, make decisions, or predict outcomes. In the standard process of machine learning, training data, data that represents prior experience in the problem domain, is inputted into a learning algorithm in order to create a model, a prediction/decision-making tool that is able to be evaluated by some performance measure. For the sake of clarity we will limit our discussion to supervised machine learning methods which are more common in TTS tasks[20].

Supervised learning is a machine learning approach where models are trained using labeled data. The training data includes input-output pairs, where the correct output for each input is provided. The model learns to predict the output from the input during training, and its performance is evaluated on unseen data. This process of "learning" is the adjustment of the weights, parameters within the model, to better generalize to the dataset in question. In this process of training, we are also given some function that will evaluate "how poorly" the model performs on the dataset (the loss function).

The goal of this process is to create a model that performs well on unseen data. Attempting to blindly lower the training error (the loss function) on the data is often counterproductive, leading to overfitting where the model memorizes the training data's noise and peculiarities rather than learning the underlying patterns. This hinders its ability to generalize to new data. The goal is to balance accurately capturing the trends in the training data while maintaining the model's flexibility to apply these learnings to new, unseen data effectively. The Fundamental Theorem of Machine Learning suggests that a model's ability to generalize (perform well on unseen data) is contingent upon striking a balance between its complexity and the training data's comprehensiveness. Too simple of a model may not capture all the underlying patterns (underfitting), while an overly complex model might learn noise from the training data (overfitting), impairing its performance on new data.

Gradient descent is an optimization algorithm used to minimize some function by iteratively moving in the direction of steepest descent, as defined by the negative of the gradient. In machine learning, this function is typically the loss function. By updating the parameters of the model in the opposite direction of the gradient of the loss function (with respect to the parameters), gradient descent seeks to find the parameter values that minimize the loss, thereby improving the model's predictions (as long as the model is not overly complex). These continual

iterations of gradient descent find minima in the loss function (given small enough step size) by iteratively bringing the function closer and closer to the minima until convergence. [20, lecture notes cis 5200]. Figure 2 visualizes this process of gradient descent, highlighting the incremental steps from the starting value in the red area to the local minima at the trough. This visualization also highlights an important issue within gradient descent, where other minima may be missed due to the lack of convexity of the loss function. For example, if the starting position were at the other trough, gradient descent would likely lead to adjustment of the parameters towards that local minimum, rather than the true global minimum, leading to a suboptimal model.

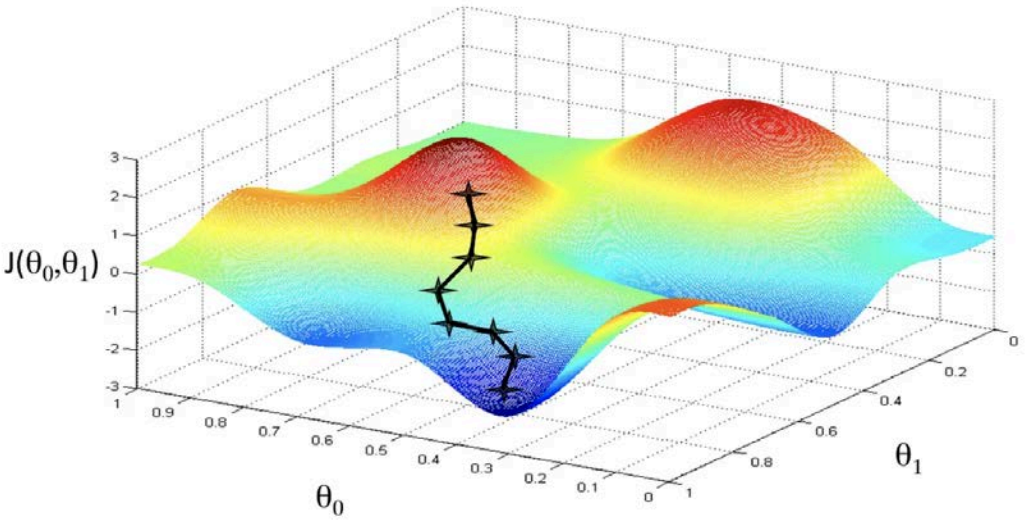

Figure 2. Visualization of Gradient Descent. The process adjust the parameters, moving opposite the direction of largest local increase[21]

Neural Networks

In a neural network model, we have multiple different layers of nodes where each layer passes off data to the next. The first layer takes in the input and sends linear combinations of the features in the data to nodes in the next layer, weighting each node in the linear combination by its weight to each of the other nodes in the next layer. The nodes in each of the middle layers, known as the hidden layers, apply an activation function, a nonlinear function, to these linear combinations and with the results of those, create linear combinations that they then send to the next layer. The choice of a nonlinear activation function is essential as otherwise, we would be taking linear combinations of linear combinations of the original input. As linear combinations of linear combinations are just linear combinations of the original input, this would thus mean that a hidden layer is redundant and would be unable to generalize to fit to an arbitrary function of the inputs. The benefit of neural networks is thus this ability to generate complex functions of the input, with each successive hidden layer finding emergent features from the input. Finally, this is

passed off to the output layer which generates the output of the model. Figure 3 shows an abstracted view of this neural network architecture, with the input layer passing off linear combinations to subsequent hidden layers (with the activation functions), culminating in the output.

Training a neural network involves two main steps. During forward propagation, input data passes through the network, layer by layer, until it produces output. The network's output is then compared to the actual target using a loss function, calculating the error. During backpropagation, this error is propagated back through the network, calculating the gradient of the loss function with respect to each weight. Finally, gradient descent updates the weights to minimize the loss, iteratively improving the network's accuracy and minimizing the loss.

Due to the nature of hidden layers and the numerous weights that may be a part of a neural network, these models will often be incredibly complex with a large number of distinct weights. As such, one might expect these models to overfit to training data. However, neural networks can effectively handle large and complex datasets, often outperforming other models in such scenarios despite their complexity. This is because despite their complexity, the training data in deep learning tasks often have an enormous sample size, meaning that the model is actually not complex enough to overfit on the training data. Moreover, these models often have loss functions that are not convex, meaning that the local minima found are not guaranteed to be global minima when performing gradient descent. Although the loss functions they use are not always convex, implying that finding the global minimum is theoretically challenging, in practice, they manage to achieve impressive results, likely as a result of their large training dataset and various techniques to find other possible local minima.

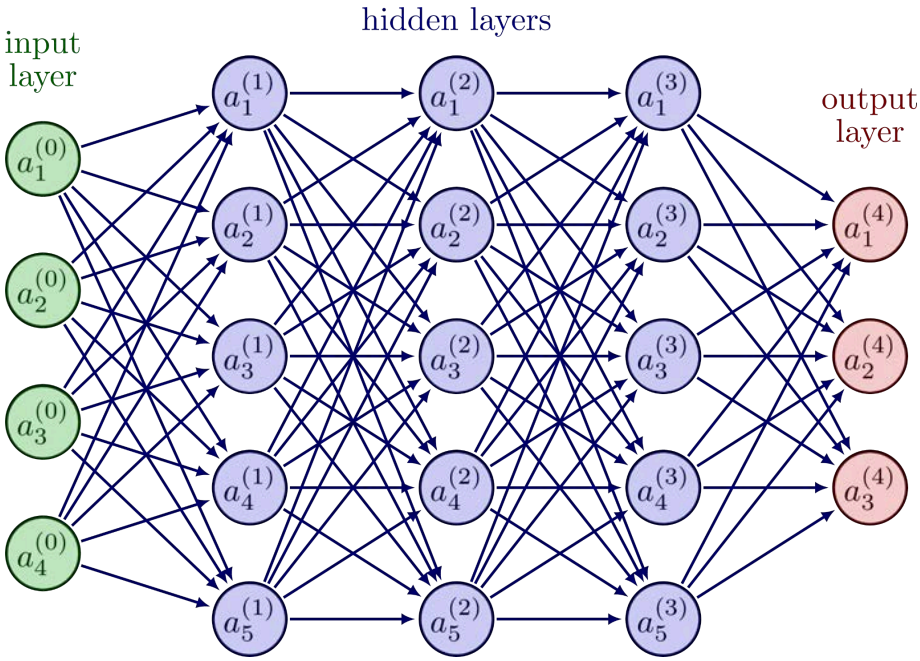

Figure 3. Basic Neural Network Architecture[22]

Recurrent Neural Networks (RNNs)

Recurrent Neural Networks (RNNs) are a class of neural networks designed to recognize patterns in sequences of data. The key difference between RNNs and other neural networks (feedforward) is that RNNs have the distinctive feature of maintaining a form of memory by using their internal state to process sequences of inputs. This means that rather than the network processing each layer in turn from the output of the prior layers, instead, the output of the model is fed back into the model at the next time step, allowing the model to use previously generated output[23]. In other words, RNNs have a loop within their structure that allows information to be passed from one step of the sequence to the next. This looping mechanism is what enables RNNs to effectively handle sequential data, making them uniquely suited for tasks where the temporal dynamics and the context of previous inputs significantly impact the current processing or prediction. By retaining a state, RNNs can incorporate the historical context of inputs into their model of the data, allowing for nuanced understanding and prediction of sequences, which is a crucial aspect in TTS where the context of syllables and words affects the manner in which they are generally spoken. We refer to the sequence of internal states as the input is processed as the “hidden states” of the model.

This ability to "remember" information about previous inputs for a short duration is achieved through the use of hidden layers. At each time step, the hidden state of the network is updated based on both the current input and the previous hidden state. This process is

mathematically represented by recurrence relations, which dictate how the hidden states evolve through time. As such, the output at any given time step can be influenced by the inputs received at preceding time steps, providing a form of short-term memory. Figure 4 shows the basic RNN architecture, where the hidden state h is updated through the input x being sequentially applied and output L is generated from each hidden state.

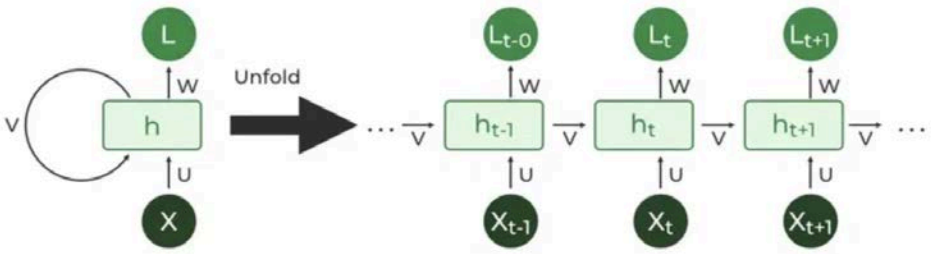

Figure 4. Recurrent Neural Network Architecture[24]

Attention

The attention mechanism allows a model to focus on certain parts of the input sequentially, parts that are deemed important for a particular task at hand. This is akin to how human attention works, focusing on parts of the input that are most relevant for understanding or decision-making. This mechanism of attention represents another method to “remember” information and the general attention mechanism is able to be used with a variety of different deep learning models[25].

Consider a model that takes an input matrix X, where X belongs to the real numbers space $R^{(dx \times nx)}$, where dx represents the dimension of the input vectors, and nx is the number of input vectors. These columns may signify various different inputs depending on input data of the model. To extract features from X, a feature model is used to produce nf feature vectors [f_1, ..., f_{nf}], where each vector belongs to R^{df}, and df denotes the size of these feature vectors. The feature model could be a range of architectures or transformations, including both deep learning models and linear transformations. This process transforms the initial input X into a series of feature vectors [f_1, ..., f_{nf}] that the attention mechanism will focus on.

The selection of feature vectors for attention is directed by a query q, belonging to R^{dq}, with dq indicating the size of the query vector. This query, formulated based on the desired output of the model, instructs the attention mechanism on which feature vectors are relevant at a

given moment. In essence, the query acts as a specific request for information from the feature vectors in relation to the current prediction context.

The attention model operates by using the feature vectors and the query as inputs. It may comprise one or several general attention modules. Each general attention module processes the query q from R^{dq} and a matrix of feature vectors $F = [f_1, ..., f_{nf}]$ from $R^{df \times nf}$. F is then used to derive two matrices: the keys matrix $K = [k_1, ..., k_{nf}]$ from $R^{dk \times nf}$ and the values matrix $V = [v_1, ..., v_{nf}]$ from $R^{dv \times nf}$, with dk and dv representing the dimensions of keys and values, respectively. The matrices K and V are typically obtained through linear transformation of F using weight matrices W_K from $R^{dk \times df}$ for K and W_V from $R^{dv \times df}$ for V.

The aim of the attention module is to compute a context vector as a weighted average of the value vectors (columns) in V. Weights are determined through an attention scoring and alignment procedure, utilizing the query q and keys matrix K to compute attention scores. These scores are then normalized through an alignment function, commonly a softmax, to sum up to one. This reflects the relative significance of each feature vector for the task at hand, resulting in a context vector c. The context vector is subsequently utilized by the output model to generate the final prediction, effectively translating the contextual insights gathered by the attention mechanism into a concrete output prediction. In doing so, models such as RNNs are generally more effective, not “forgetting” earlier parts of the input sequence, a problem typically associated with RNNs due to earlier parts of the input sequence having a diminished effect on the output of the model.

Self-attention is a specific instance where the model focuses on different parts of a single sequence, rather than on different sequences or inputs. In self-attention, the query, key, and value vectors are all derived from the same input sequence, allowing the model to weigh the importance of different parts of the input with respect to each other. In particular, the query may be constant, such as in classification tasks with a singular output, or learned, by calculating the query as a function of a weight matrix and the feature vectors. This is particularly useful in tasks where understanding the relationship or the relevance of different parts of the input to each other is crucial, such as the text-to-speech case. Moreover, this process encodes information about the relations between individual words such as how the presence of one word may affect the pronunciation and tone of other words.

Attention Is All You Need (Transformers)

The Transformer model represents a significant shift in how sequence-to-sequence tasks are approached in the field of deep learning. While the attention mechanism had previously been used within RNNs to improve their ability to handle long-distance dependencies, the Transformer model removes recurrent layers entirely, relying solely on attention mechanisms to process sequences. This design choice not only enhances the model's ability to capture complex

relationships across different parts of the sequence but also significantly increases computational efficiency and training speed. This is due to the lack of recurrence, allowing for parallelization of computation due to a lack of time-dependent steps of output computation (the model does not have to process the input sequence one by one)[1].

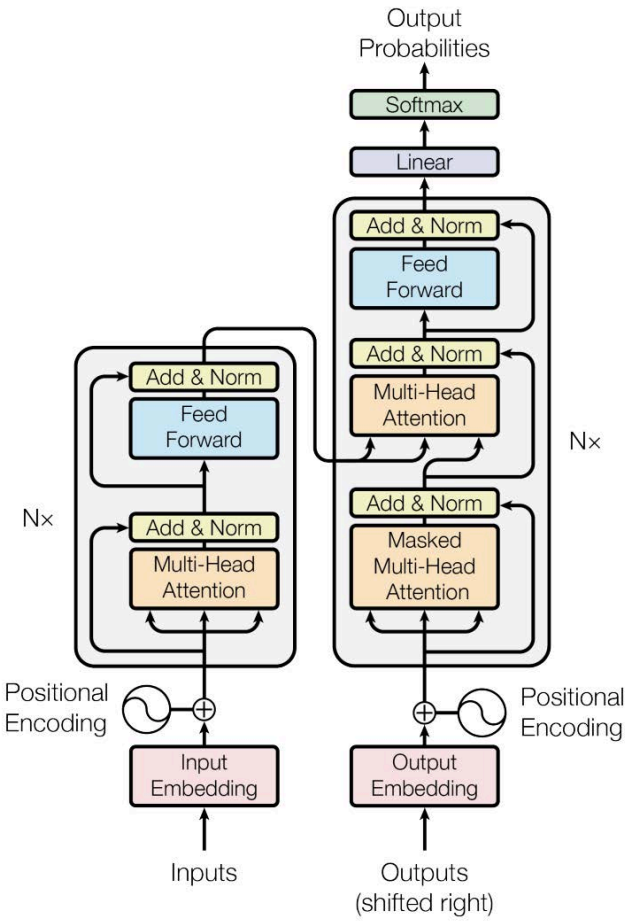

Figure 5. Transformer Architecture [1]

Perhaps the most critical innovation in the Transformer model is the concept of multi-head attention. This mechanism allows the model to focus on different parts of the sequence simultaneously. By projecting the queries, keys, and values multiple times with different, learned linear projections to dk, dk, and dv dimensions, respectively, the model can capture various aspects of the information in different subspaces. After performing the attention function in each of these projected versions of queries, keys, and values independently, their outputs are concatenated and once again linearly transformed. This process enhances the model's ability to focus on different positions, making it more flexible and powerful compared to the single-head attention mechanism. Figure 5 shows the general transformer architecture as initially

proposed, generating the output from neural networks and attention mechanisms both in the input and output.

Encoder-Decoder Architecture

The encoder-decoder architecture is designed to process sequential input data, transforming it into a different output sequence. This is essential in TTS, where we may have variable length inputs and variable length outputs, highlighting the importance of some mechanism to “collapse” the information in the input before beginning to generate an output. Thus, the encoder processes the input sequence to create a context vector, capturing the essence of the input data. The decoder then uses this context vector to generate the output sequence[26].

Figure 6. Basic Encoder-Decoder Architecture[27]

The basic encoder-decoder architecture is highly flexible, allowing for different types for models and attention both within the encoder and decoder and also between them. In an RNN encoder-decoder, within the encoder, an RNN can capture the context of the input sequence, which is then condensed into a fixed-size context vector representing the entire input sequence's information. This vector serves as the initial state for the decoder RNN, which generates the output sequence step by step, using its “memory” of what has been generated so far to inform the next output. Extensions of this baseline model utilizing attention may also include self-attention mechanisms in the encoder to learn feature representations of the input as well as self-attention mechanism in the decoder to attend to the positions in the output. Moreover, attention between the two may be incorporated by using output from the encoder (such as the hidden states) to construct the keys and values while using output from the decoder (such as the current hidden state) as the query at a given moment[26][1]. Figure 6 shows a general abstraction of the encoder-decoder architecture, encoding information from the input to generate some state and then using a decoder to generate output based on the “memory” of the input and previously generated output.

In the transformer model, the encoder consists of a stack of identical layers, each containing two sub-layers. The first sublayer is a multi-head self-attention mechanism, and the second is a simple, position-wise fully connected feed-forward (non-recurrent) network. A key feature here is the use of positional encoding, added to the input embeddings to provide some information about the position of the tokens in the sequence since the model itself does not incorporate recurrence. Furthermore, in this model, the decoder is also composed of a stack of identical layers. However, in addition to the two sub-layers found in the encoder layers, each

decoder layer has a third sub-layer, which performs multi-head attention over the encoder's output. This setup allows every position in the decoder to attend to all positions in the encoder sequence[1].

GANs

Generative Adversarial Networks (GANs) represent a powerful class of deep learning models, facilitating innovative approaches in generating realistic, high-quality synthetic data through the adversarial training of generator and discriminator networks. At the heart of this framework lies the Generator (G), whose primary function is to adeptly capture the underlying data distribution. Operating in tandem, yet in contention, is the Discriminator (D), a model tasked with the critical role of discerning whether a piece of data originates from the authentic data distribution or is an artifact of G's generated distribution[28][29].

This interaction is formalized through a minimax (zero-sum) game, where both models engage in a competition leading to a Nash equilibrium, facilitated by a simultaneous Gradient Descent. At this equilibrium, the Generator produces data indistinguishable from the true distribution to such an extent that the Discriminator cannot reliably differentiate between authentic and generated instances. The updating mechanisms for both G and D are informed by gradient signals derived from the loss, which quantifies divergences between the two distributions as assessed by the Discriminator[28]. During this process, G does not have direct access to real training data (such as the speech of a real individual), instead learning through this interaction with the discriminator. Meanwhile, D is given access to both the generated examples and the real training data and must attempt to distinguish these. Through D's error on distinguishing these, calculated by labeling the examples with whether they are generated or real, we are thus able to train D, attempting to minimize this error. Similarly, we can use this error to train G, attempting to maximize this error[30]. Figure 7 highlights the architecture of this model, with the generator creating fake samples and the discriminator taking those samples along with real samples in order to generate predictions of genuineness to further update the parameters of the model.

More formally, G takes a noise vector z, usually sampled from a Gaussian distribution, as input and generates data G(z)[28]. The function of the Generator can be mathematically represented as: $G(z; \theta_g)$ where θ_g denotes the parameters (weights) of the Generator. Similarly, D takes in data x and outputs a probability D(x), representing the likelihood that x comes from the real data distribution rather than from the Generator. The function of the Discriminator is given by $D(x; \theta_d)$. The value function V(D, G) that both D and G aim to optimize and corresponding minimax problem is defined by

$$\min\theta_g \max\theta_d V(D, G)$$

with

$$V(D,G) = E_{x\sim p\_data(x)}[\log D(x)] + E_{z\sim p\_z(z)}[\log(1 - D(G(z)))]$$

where the first term, $E_{x\sim p\_data(x)}[\log D(x)]$, represents the expected log-probability that D correctly identifies real data x as real and the second term, $E_{z\sim p\_z(z)}[\log(1 - D(G(z)))]$, represents the expected log-probability that D correctly identifies generated data G(z) as fake. The choice of log-probability is used because it provides useful mathematical properties for optimization, such as numerical stability and easier gradient descent (due to the conversion of the product of probabilities to the sum of log-probabilities). One issue with this basic approach is that the second term of the objective function often leads to a vanishing gradient problem, where once the discriminator becomes better and better at classification, D(G(z)) tends towards 1, causing log(1-D(G(z)) to yield large negative values. As such, the gradient tends to very small magnitudes, causing the update during gradient descent to be minimal, halting the learning process due to slow updates to the parameters. Due to these practical issues, the problem is reformulated to the two objectives

$$\max\theta_D \, E_{x\sim p\_data(x)}[\log D(x)] + E_{z\sim p\_z(z)}[\log(1 - D(G(z)))]$$

and

$$\max\theta_G \, E_{z\sim p\_z(z)}[\log(D(G(z)))]$$

which critically have the same fixed points (points of equilibrium) for G's gradient and train in the same direction.

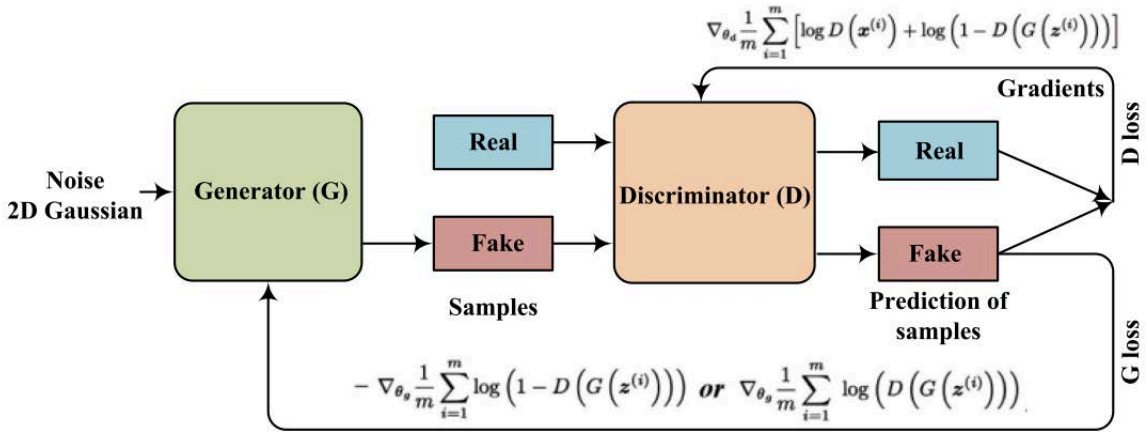

Figure 7. Simplified GAN architecture[28]

Conditioning refers to the inclusion of additional information to direct the data generation process, thus creating a Conditional GAN (cGAN). This supplementary information can be anything relevant to the task, such as class labels in image generation or textual descriptions for text-to-image synthesis. By feeding this additional context into both the Generator and the Discriminator, cGANs are able to produce outputs that are not just random samples from the learned data distribution, but are aligned with the given condition[31]. More formally, this involves modifying both the Generator and Discriminator to accept additional input in the form of a conditioning variable y. This variable y could represent any kind of auxiliary information

such as class labels or data from other modalities. This is critical in TTS, where by conditioning on text, a GAN may incorporate the information from the input text in its output, thus allowing for the GAN to create output tailored to the input. Thus, rather than generating arbitrary audio, we may actually create TTS models.

Diffusion

Denoising Diffusion Probabilistic Models (DDPMs) represent a class of generative models that have gained popularity due to their ability to generate high-quality samples, similar to those produced by GANs, but with a different underlying mechanism. DDPMs are inspired by the process of diffusion, which is a physical process that describes the movement of particles from regions of high concentration to regions of low concentration. In the context of DDPMs, this process is modeled in a probabilistic framework to gradually convert data into a simple distribution (e.g. Gaussian noise) and then learn to reverse this process to generate new data samples[32].

The diffusion process in DDPMs is defined as a Markov chain of conditional distributions that gradually adds noise to the data over a sequence of time steps T. The process starts with the data distribution $q(x_0)$ and applies a series of small Gaussian noise perturbations to reach a distribution close to Gaussian noise $q(x_T)$. This is mathematically represented as:

$$q(x_t \mid x_{t-1}) = N(x_t; \text{sqrt}(1-\beta_t) * x_{t-1}, \beta_t * I)$$

where x_t denotes the data at time step t, β_t is a small positive value controlling the amount of noise added at each step, and N denotes the normal distribution.

The reverse process aims to learn to denoise, i.e., to generate samples from the noise distribution $q(x_T)$ back to the data distribution $q(x_0)$ through a series of learned reverse transitions. This is represented as:

$$p_\theta(x_{t-1} \mid x_t) = N(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$

The goal of this process is to minimize the Kullback-Leibler (KL) divergence between the true posterior distribution of the reverse process and the model's approximation of it. The KL divergence is a measure of how a distribution differs from a reference distribution. In particular, we utilize the variational bound, a lower bound on the log-likelihood of the observed data under the model. Maximizing this bound allows us to indirectly maximize the log-likelihood, which is our ultimate goal in generative modeling but is often too computationally intensive to optimize directly[32].

Although this has the natural application in TTS through conditioning the model on the input sequence and attempting to predict the audio output, diffusion models have made significant strides in enhancing audio quality, proving effective in text-to-speech generation as well as restoring degraded audio. Audio degradation can result from factors like noise and reverberation, necessitating methods to clean the original signal or fill in missing information.

There are two main approaches to audio restoration: discriminative methods, which aim to minimize the difference between the enhanced and original clean audio, and generative models, which estimate the distribution of clean signals. Discriminative methods, while achieving impressive objective metrics, often lack the naturalness that generative models provide. Diffusion models, which fall into the generative category, offer a promising way to bridge this gap by leveraging their unique ability to capture the distribution of clean audio signals, leading to more natural-sounding speech enhancement. This is in turn essential due to the issues of noise in training datasets for speech synthesis as compiling clean data is often essential to ensure the resulting model does not pick up this noise in its output[8].

Three and Two Stage Frameworks

A traditional Text-to-Speech (TTS) system typically involves a three-stage framework. First, there is text analysis, where the text input is processed to extract linguistic features, such as phonemes, prosody, and syntax. This stage often involves linguistic analysis and text normalization. Then, in the acoustic model, the extracted linguistic features are then mapped to corresponding acoustic features (like mel-spectrograms). Finally, a vocoder synthesizes audio waveforms from the acoustic features to produce the final speech output. This framework has been foundational in traditional TTS systems, providing a structured approach to converting text into speech. However, it often requires extensive tuning and expert knowledge to optimize each stage[8].

A simplified version, the two-stage framework, merges steps of this framework, utilizing deep learning to enable for complex relationships between the input and output sequences. In particular, the most widespread two-stage framework utilizes a deep neural network in order to generate mel-spectrograms directly from the text, merging the linguistic analysis and acoustic feature generation into one model. Once the mel-spectrograms have been generated, a vocoder can be used on the acoustic features generated, much like the three-stage framework, in order to create the speech output. This is the primary framework used in diffusion models and is generally done using two separate deep learning models, one for each stage. Other two-stage frameworks exist, such as a model that generates linguistic features in the first stage and then utilizes the final audio waveform directly from the linguistic features[8].

Acoustic Model

Acoustic modeling is a crucial component in many TTS systems, employing statistical and computational techniques to transform text into its corresponding sound signals[8]. These sound signals, most often mel-spectrograms, serve as intermediate representations that capture the nuances of human speech. The acoustic model bridges the gap between the text and the final waveform, acting as the main predictor of the audio characteristics that the vocoder will later synthesize into natural speech. In traditional TTS systems, acoustic models rely on carefully engineered features and handcrafted rules to map linguistic features to acoustic features.

However, with the advent of deep learning, neural networks have become the preferred choice for acoustic modeling, as they excel at learning complex mappings directly from data. Models like Tacotron and FastSpeech employ sequence-to-sequence architectures, including attention mechanisms, to better capture the temporal dependencies between text and speech. This leads to more accurate and natural-sounding synthesized speech[33]. Moreover, controllable emotional models are crucial for adding expressive and natural emotional tones to synthesized speech, yet they often face challenges in precisely measuring emotion intensity values without compromising audio quality. As such, models such as EmoDiff have been created which first train an unconditional acoustic model and then an emotional classifier along the diffusion trajectory. During inference, the classifier's soft labels guide the generated audio, resulting in high-quality speech with precisely controlled emotion, with experimental results affirming that EmoDiff successfully synthesizes emotionally expressive speech without sacrificing audio fidelity[34].

Diffusion models often require hundreds or thousands of iterations to generate high-quality audio. As such, knowledge distillation is often used in order to create faster models with similar performance. Knowledge distillation is a technique where a smaller, more efficient model (student) learns to mimic the performance of a larger, more complex model (teacher). This approach is often used to transfer knowledge from a well-performing but computationally expensive model to a lighter one that can be used in environments with limited resources. The teacher model is trained using a standard training process. In the case of an acoustic model, this may be a diffusion model that generates high-quality mel-spectrograms from text. Then, the student model is initialized, often with fewer parameters or a simpler architecture, making it more suitable for fast inference. The student model learns by trying to replicate the outputs of the teacher model. This is usually done by minimizing the difference between the student's predictions and the teacher's predictions for the same input data. The student may also learn from the intermediate representations and activations of the teacher model. The distillation process helps the student model learn to generate the same high-quality results as the teacher model, but with fewer diffusion steps. This reduces the computation time significantly while maintaining the quality of the generated audio.[35]

Combining the technologies of GANs and diffusion, modern models have been able to achieve fast, high quality speech synthesis. For example, the acoustic model in DiffGAN-TTS is implemented as a diffusion decoder that learns to generate Mel-spectrograms for speech synthesis by gradually denoising the input data. Unlike traditional autoregressive models, which suffer from slow synthesis speeds and errors in text-to-speech alignment, diffusion-based acoustic models like DiffGAN-TTS adopt a non-autoregressive approach, leveraging a powerful acoustic generator trained adversarially to approximate the denoising distribution. This method allows for larger denoising steps during inference, significantly accelerating the sampling process while maintaining high-quality speech output. A key feature of DiffGAN-TTS is its

two-stage training process, where the first stage trains a basic acoustic model to provide prior knowledge. In the second stage, this basic model helps refine the acoustic generator by conditioning the diffusion model on coarse Mel-spectrogram predictions. This ensures that the generator accurately reconstructs the desired acoustic features, providing high-quality, multi-speaker TTS performance. This demonstrates the ability of such models to generate high-fidelity speech with as few as one denoising step, marking a significant leap in acoustic model performance for TTS systems[36].

Vocoder Model

Neural Vocoders play an essential role in Text-to-Speech systems, transforming acoustic models into audible speech by accurately synthesizing the human voice. Vocoder models transform acoustic features, such as Mel-spectrograms, into audio waveforms. In the field of neural vocoders, autoregressive models have historically been popular due to their high-quality output, but they have low inference speed. Non-autoregressive models improved inference speed, yet initially struggled to match the audio quality of their autoregressive counterparts. WaveGrad introduced a new direction in neural vocoders by employing score matching and diffusion models to estimate the gradient of the data log-density. It predicts the trajectory of the audio waveform by refining the noise distribution through several steps, each improving the waveform's resemblance to natural speech. This stepwise process results in high-quality audio while being computationally efficient[37]. Similarly, improvements on this baseline model have been proposed and implemented, utilizing noise generated from distributions such as the gamma distribution rather than gaussian noise. This choice is made due to the fact that gaussian noise may be a poor representation of the different segments of the audio samples, leading to training inefficiency due to a mismatch in the actual data distribution and the noise that the model attempts to denoise during inference[38].

End-to-End Frameworks

End-to-End frameworks in Text-to-Speech synthesis streamline the conversion from text to audible speech by integrating multiple processing stages into a unified, coherent model, significantly enhancing efficiency and naturalness. End-to-end frameworks thus shift away from treating acoustic and vocoder modeling as separate processes. Initially, partially end-to-end methods combined acoustic modeling and vocoder training in a joint manner, yet they still maintained separate models for each stage. Fully end-to-end frameworks, however, utilize a single model to generate speech directly from text, bypassing explicit intermediate acoustic features. Notable fully end-to-end models like FastSpeech 2 and EFTS-Wav incorporate adversarial decoders or GANs in order to do this end-to-end task. While most end-to-end approaches still rely on mel-spectrograms for text-to-speech alignment, these end-to-end frameworks continue to advance, particularly with the integration of diffusion models, promising further improvements in synthetic speech naturalness and quality[8]. Notably, the reason for this end-to-end training is due to improvements in the internal representations of the data within the

model. By allowing the model itself to learn the intermediate representations of the data such as the acoustic and linguistic features, the model can optimize these representations, leading to a more accurate model that is free from more human biases. Moreover, the model learns to align text with audio internally, eliminating the need for pre-aligned data. Thus, with all components integrated, errors don't propagate between discrete stages[39].

Ethical Considerations in Modern TTS Technologies

Audio Deepfakes

The proliferation of audio deepfakes highlights critical ethical and security concerns, as they enable the creation of persuasive yet fraudulent audio clips indistinguishable from genuine human speech. This emerging technology poses serious threats, particularly in disinformation and exploitation. Audio deepfakes leverage deep learning techniques such as GANs, Convolutional Neural Networks (CNNs), and Deep Neural Networks (neural networks with a hidden layer). These technologies have enabled rapid advancements in voice cloning systems, which can learn speech characteristics from limited samples and produce convincing, fabricated speech[40]. However, this level of sophistication introduces new security risks.

Audio deepfakes have already been used maliciously, including instances where cybercriminals cloned voices to commit fraud and manipulation. For example, deepfake audio technology was utilized to mimic the voice of a German executive, resulting in the transfer of €220,000 to a fraudulent account. Moreover, attempts to manipulate public opinion have also emerged, such as an audio deepfake of President Biden that was used to spread misinformation, urging people not to vote in the New Hampshire primary. The creator of this deepfake was suspended by ElevenLabs, the company whose technology was used to create this synthetic speech, but the widespread availability of such technology represents a critical issue that blurs the line between real and artificially generated speech[41].

Audio Deepfake Detection

While audio deepfakes represent a remarkable feat in speech synthesis, they underscore the pressing need for robust detection techniques to combat their potential for disinformation and exploitation. Existing detection methods focus on features like Mel-Frequency Cepstral Coefficients (coefficients derived from spectrograms), spectral features, and others to distinguish between authentic and synthetic audio. CNN architectures like VGG-16 have proven effective in detecting audio deepfakes, leveraging spectrograms and MFCCs to classify real and fake audio clips[42]. Short-term spectral features provide detailed information on the signal's spectral content over a short period, which is essential in identifying subtle anomalies introduced during synthesis, while long term and prosody features are used to detect anomalies present in longer phrases and utterances. Moreover, due to the limitation of these more handcrafted features implicitly encoding biases, deep learning features, generated through the use of neural networks, are often used in order to assist generalize detection models across various datasets[43][44]. Similarly, attention mechanisms have been utilized in detection models in order to learn the relationships between audio cues that vary across time and frequency, allowing for more robust audio deepfake detection.

Diversity of data

Data diversity is crucial for creating inclusive and adaptable speech models that accurately represent the nuances of different languages, dialects, and personal speech patterns. The collection of such data represents a central problem for both speech synthesis and speech synthesis detection, both from a logistical and legal perspective. A comprehensive dataset must represent various age groups, genders, and dialects to ensure realistic scenarios for model training. This requires curated collections from diverse sources, ensuring each demographic is sufficiently represented. Furthermore, data collection must adhere to global privacy laws like the European General Data Protection Regulation (GDPR) and the US-state California Privacy Rights Act (CCPA). This involves obtaining explicit consent from speakers, anonymizing data, and respecting usage restrictions. Sourcing audio from publicly available platforms without appropriate permissions risks legal consequences, meaning that much of Internet scraping of data leads to datasets unusable by the broader research community. From a technical perspective, this data collection is also difficult, due to the issue of differences in recording equipment, leading to the possible issue of learning the noise of the equipment rather than the more fundamental distribution of real audio data[44].

Federated Learning (FL) has emerged as a response to increasing data privacy concerns and regulations, enabling collaborative training across decentralized data centers and devices while safeguarding data privacy. By training models without exchanging raw data, FL addresses privacy issues but introduces new challenges due to the heterogeneous distribution of data across network nodes. This non-independent and identically distributed (non-IID) data reduces the learning effectiveness of FL compared to centralized training methods as the model is not able to simply learn the distribution. Though various approaches have been proposed to tackle this issue, it remains an open research question. FL offers a promising pathway for developing robust models capable of detecting synthetic speech by streamlining compliance with data regulations. However, achieving industrial-grade performance in this context requires addressing the non-IID data challenge, particularly in the audio domain. An open-source dataset that simulates a distributed learning environment with diverse synthetic and real audio data could provide the foundation for this research. Such a dataset would need to include varied TTS algorithms and real audio samples that reflect a broad demographic, both phonetically and dialectically[44].

Technical Challenges and Commercial Applications

Technical Challenges

Deep learning based speech synthesis models overcome the limitations of prior models by leveraging complete context information, using deep neural networks to map context features to high-dimensional acoustic features. This results in better quality than traditional methods; however, these powerful models introduce new challenges. The need for numerous hidden layers and nodes increases the network's parameter count, resulting in higher time and space complexities for training. This demands significant computational resources and large datasets, making training expensive and prone to overfitting when data is insufficient. Moreover, collecting such data is costly and time-consuming. Research into semi-supervised or unsupervised training can help address this by leveraging unpaired text and speech data, which are more readily available. Similarly, deep neural networks necessitate extensive computations, making parallelization essential. Implementing this parallelization involves either multi-machine setups or GPU parallelization; however, GPU code development can be complex, necessitating collaboration between hardware and software vendors to create intelligent tools that streamline programming and enhance network efficiency. From a more algorithmic perspective, End-to-end TTS models achieve state-of-the-art performance but face challenges in front-end text analysis, which is crucial for understanding context and linguistic features. More research is needed to improve the handling of context in end-to-end systems to bridge the gap between text and speech effectively[45].

One of the primary issues is the accurate expression of emotion and intonation, both of which are critical for natural-sounding speech but are complex to model due to their highly nuanced nature. Emotion in TTS involves the dynamic modulation of pitch, volume, and rhythm, which current models struggle to replicate authentically. Furthermore, deep language understanding and commonsense reasoning are essential for high-quality TTS. This includes correctly interpreting and pronouncing acronyms, numbers, and handling code-switching—where speakers switch between languages or dialects mid-sentence—as well as understanding humor and conversational speech[46][47][48]. These challenges have been addressed by modern models, with varying degrees of success, though the need to account for the multitude of use cases and diversity of language represents a significant barrier. Moreover, while TTS technology has advanced, surpassing the naturalness and style of prior models, it is not yet on par with human narrators, such as those on platforms like Audible, particularly when it comes to nuanced and expressive content.

Creating representative synthetic audio data for audio deepfake detection also presents significant challenges. The data needs to be carefully paired with corresponding real speech to

maintain the balance found in natural voices. This ensures that synthetic data closely reflects real-world conditions, providing a realistic benchmark for detection models. The variability of audio synthesis components like vocoders also poses a challenge, as they respond differently depending on their training. Vocoders trained on multi-speaker datasets are often of the higher quality when synthesizing speech using a voice not in the training dataset, while those trained on single-speaker datasets may have artifacts that we would not want our detection mechanisms to pick up. Similarly, Vocoders fine-tuned for specific voices, despite being resource-intensive to train, produce highly realistic audio that represents an open problem within audio deepfake detection. TTS feature extraction and voice conversion models must also be evaluated under different conditions, such as direct application of pre-trained single-speaker models and fine-tuned usage on both high and low-quality speaker data. Ensuring consistency across these varied conditions while covering different vocoder combinations introduces complexity. Achieving natural prosody is crucial for synthetic speech to closely mimic real speech, and datasets must include longer sentences to capture the subtlety of expressive intonation and minimize artificial patterns that can expose synthetic origins[44].

The rapid advancements in neural networks and their application in various critical fields like computer vision and natural language processing have increased the need for explainability in AI systems. The European Union's "right to explanation" emphasizes the importance of transparent AI, particularly for overseeing and regulating the use of AI technologies. This necessitates methods that unravel the decision-making process of black-box models, leading to the development of Explainable Artificial Intelligence (XAI) methods. Techniques like Grad-CAM, which provide visual insights into the functioning of CNNs, are widely used in image classification but struggle with audio data visualizations due to ambiguities in spectrogram representations. Therefore, developing novel XAI algorithms tailored for audio classification tasks is crucial. These methods can uncover the distinctive voice characteristics that separate synthetic and real speech, improving model generalization and guiding the design of feature-rich datasets. Such advances will aid in building robust synthetic speech detection models, enabling better model interpretability and fostering trust in AI technologies[44].

To combat the unpredictable nature of spoofing attacks, it is crucial to develop algorithms that generalize well to unseen scenarios. For replay attacks, where one’s voice is recorded and played back, combining traditional audio features with complex classifiers has proven to be highly effective. Analyzing the ASVspoof challenge, where participants work to create a model to detect audio deepfakes, reveals a common trend: participants often optimize their models for the provided dataset, inadvertently causing overfitting. This becomes evident when models exhibit significant performance drops on novel data with varied characteristics and pre-processing methods. To enhance the generalization capability of synthetic speech detection models, research must extend beyond conventional audio feature extraction and focus on novel neural network architectures. Additionally, exploring new audio features that don't solely rely on

specific vocal characteristics (such as pitch and timbre) can help models learn more generalized information, ultimately improving the detection of synthetic speech in diverse conditions[44].

Adversarial Data Augmentation (A-DA) is an innovative approach that aims to enhance the robustness of deep speaker models, those which try to classify who is speaking in an audio sample, against acoustic variations. Traditional data augmentation methods, which are popular for enriching training data by simulating real-life acoustic variations, can introduce unwanted distortion, known as augmentation residuals, that reduces model generalizability. To address this, A-DA incorporates adversarial learning by combining data augmentation with an additional augmentation classifier. This classifier categorizes different augmentation types and uses a gradient reversal layer during backpropagation to make speaker embeddings more resilient to these augmentation variations. This approach helps the network learn speaker representations that can deceive the augmentation classifier, ultimately leading to speaker embeddings that are more robust to various acoustic conditions. Experiments conducted using datasets like VoxCeleb and CN-Celeb demonstrated that A-DA consistently outperforms standard data augmentation methods, even in challenging augmentation conditions, proving its effectiveness in improving robustness and generalization[49].

Commercial Applications

Audiobooks, Accessibility Tools, Customer Service and Support, Healthcare and Medical Devices all represent fruitful uses for these modern TTS systems, specifically benefitting from their accuracy and naturalness. In the healthcare domain, for instance, TTS systems have played a crucial role in developing speech synthesis technologies for individuals with vocal disabilities, enabling the creation of personalized voices. Such personalized voices also help in accessibility tools for people with speech impairments, enhancing social interaction and overall quality of life. Similarly, the ability to quickly generate audiobooks at a computation cost rather than hiring voice actors represents a massive opportunity in the commercial space. [50][51][52].

Translation is an exciting new use of these technologies, with modern models able to directly translate one's own voice into different languages, even in the zero-shot context. The development of models like VALL-E X introduces advanced cross-lingual neural codec language modeling, allowing for high-quality zero-shot cross-lingual text-to-speech synthesis. This model can accurately predict target language acoustic tokens using the source language speech and target language text as prompts, maintaining the speaker's unique voice, emotion, and acoustic characteristics. This capability opens up possibilities for real-time voice translation applications in various sectors, like customer service and international communication, making communication more seamless across language barriers[53].

Conclusion

Text-to-Speech models have undergone drastic changes in recent years, with both academia and industry continuing to rapidly iterate and improve on these new models. The shift from traditional concatenative synthesis to advanced deep learning techniques like Generative Adversarial Networks (GANs) and diffusion models have enhanced the quality and naturalness of synthesized speech, enabling models to produce output that closely resembles human speech. This evolution has been accompanied by various architectures, including encoder-decoder models and attention mechanisms, which have significantly improved the way these models handle sequential data. These models have shown remarkable success in overcoming the robotic and unnatural outputs of earlier systems, providing more dynamic and context-aware speech synthesis. Moreover, the integration of advanced neural network architectures has opened new avenues for realistic and expressive speech generation, which can be tailored to specific emotions and nuances.

The challenges in this field are multifaceted, involving ethical considerations like audio deepfakes, technical challenges in data collection and computational resource requirements, and the need for explainability in these complex models. With these advancements come new challenges and responsibilities. Ethical considerations, particularly concerning the misuse of TTS technologies such as audio deepfakes, have emerged as critical areas requiring vigilant oversight and innovative solutions. The development of robust detection mechanisms and ethical guidelines will be crucial in mitigating risks associated with these technologies. Despite these challenges, TTS technologies hold immense promise across various domains, from accessibility tools to healthcare and global communication.

Looking forward, the TTS field is likely to focus on developing models that are not only more efficient and accurate but also ethically responsible. The integration of these models into commercial applications will continue to expand, necessitating further research into improving model generalization, robustness, and ethical safeguards. Continued research and collaboration between technologists, ethicists, and policymakers will be essential in harnessing the full potential of TTS technologies while safeguarding against potential abuses. By addressing the current limitations and leveraging the latest advancements in machine learning, TTS systems are poised to revolutionize the way humans interact with technology.

References

[1] A. Vaswani et al., “Attention is all you need,” arXiv.org, https://arxiv.org/abs/1706.03762 (accessed Mar. 28, 2024).

[2] Professor Ewan Birney and Professor John McGeehan, “Alphafold,” Google DeepMind, https://deepmind.google/technologies/alphafold/ (accessed Mar. 28, 2024).

[3] P. P. Ray, “CHATGPT: A comprehensive review on background, applications, key challenges, bias, ethics, limitations and future scope,” Internet of Things and Cyber-Physical Systems, https://www.sciencedirect.com/science/article/pii/S266734522300024X (accessed Mar. 28, 2024).

[4] P. Esser et al., “Scaling rectified flow transformers for high-resolution image synthesis,” arXiv.org, https://arxiv.org/abs/2403.03206 (accessed Mar. 28, 2024).

[5] J. Manyika and J. Bughin, “The coming of AI Spring,” McKinsey & Company, https://www.mckinsey.com/mgi/overview/in-the-news/the-coming-of-ai-spring (accessed Mar. 28, 2024).

[6] N. Kaur and P. Singh, “Conventional and contemporary approaches used in text to speech synthesis: A Review - Artificial Intelligence Review,” SpringerLink, https://link.springer.com/article/10.1007/s10462-022-10315-0 (accessed Mar. 28, 2024).

[7] Y. A. Li, C. Han, V. S. Raghavan, G. Mischler, and N. Mesgarani, “StyleTTS 2: Towards Human-Level Text-to-Speech through Style Diffusion and Adversarial Training with Large Speech Language Models,” arxiv,

[8] C. Zhang et al., “A survey on audio diffusion models: Text to speech synthesis and enhancement in Generative AI,” arXiv.org, https://arxiv.org/abs/2303.13336 (accessed May 5, 2024).

[9] A. Wali et al., “Generative adversarial networks for Speech Processing: A Review,” Computer Speech & Language, https://www.sciencedirect.com/science/article/pii/S0885230821001066 (accessed Mar. 28, 2024).

[10] A. Natsiou and S. O’Leary, “Audio representations for deep learning in Sound synthesis: A review,” ar5iv, https://ar5iv.labs.arxiv.org/html/2201.02490 (accessed Mar. 28, 2024).

[11] A. A. Abdulsatar, V. V. Davydov, V. V. Yushkova, A. P. Glinushkin, and V. Y. Rud, "Age and gender recognition from speech signals," *Journal of Physics: Conference Series*, vol. 1410, no. 1, p. 012073, Dec. 2019. doi:10.1088/1742-6596/1410/1/012073

[12] L. Sheng, D.-Y. Huang, and E. N. Pavlovskiy, "High-quality speech synthesis using super-resolution Mel-Spectrogram," arXiv.org, https://arxiv.org/abs/1912.01167 (accessed Mar. 28, 2024).

[13] T. Giannakopoulos and A. Pikrakis, "Signal transforms and filtering essentials," *Introduction to Audio Analysis*, pp. 33–57, 2014. doi:10.1016/b978-0-08-099388-1.00003-0

[14] V. Tiwari, MFCC and its applications in speaker recognition, https://www.researchgate.net/publication/265809116_MFCC_and_its_applications_in_speaker_recognition (accessed Mar. 28, 2024).

[15] S. Young, J. Odell, D. Ollason, V. Valtchev, and P. Woodland, "5.4 Filterbank Analysis," in *The HTK Book*, 2.1.,

[16] C. Tralie, "Mel Filterbanks and Mel Spectrograms," CS472A, https://ursinus-cs472a-s2021.github.io/Modules/Module16/Video4 (accessed May 5, 2024).

[17] R. A. Khan and J. S. Chitode, "Concatenative Speech Synthesis: A Review," *International Journal of Computer Applications*, vol. 136, no. 3, pp. 1–6, Feb. 2016. doi:10.5120/ijca2016907992

[18] A. Indumathi and E. Chandra, "Survey On Speech Synthesis," *Signal Processing: An International Journal*, vol. 6, no. 5, 2012.

[19] T. Raitio *et al.*, "HMM-based speech synthesis utilizing glottal inverse filtering," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 1, pp. 153–165, Jan. 2011. doi:10.1109/tasl.2010.2045239

[20] J. Gardner, "Lecture Notes" presented in CIS 5200 (Machine Learning), Fall 2023, University of Pennsylvania, Philadelphia, PA

[21] J. Adejumo, "Gradient descent from scratch- batch gradient descent, stochastic gradient descent, and Mini-Batch...," Medium, https://medium.com/@jaleeladejumo/gradient-descent-from-scratch-batch-gradient-descent-stochastic-gradient-descent-and-mini-batch-def681187473 (accessed Mar. 28, 2024).

[22] I. Neutelings, "Neural networks," TikZ.net, https://tikz.net/neural_networks/ (accessed Mar. 28, 2024).

[23] O. I. Abiodun *et al.*, “State-of-the-art in Artificial Neural Network Applications: A survey,” *Heliyon*, vol. 4, no. 11, Nov. 2018. doi:10.1016/j.heliyon.2018.e00938

[24] “Introduction to recurrent neural network,” GeeksforGeeks, https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/ (accessed May 5, 2024).

[25] G. Brauwers and F. Frasincar, “A general survey on attention mechanisms in deep learning,” arXiv.org, https://arxiv.org/abs/2203.14263 (accessed Mar. 28, 2024).

[26] K. Aitken, V. V. Ramasesh, Y. Cao, and N. Maheswaranathan, “Understanding how encoder-decoder architectures attend,” arXiv.org, https://arxiv.org/abs/2110.15253 (accessed Mar. 28, 2024).

[27] The Encoder–Decoder Architecture, https://d2l.ai/chapter_recurrent-modern/encoder-decoder.html.

[28] D. Saxena and J. Cao, “Generative Adversarial Networks (GANs),” *ACM Computing Surveys*, vol. 54, no. 3, pp. 1–42, May 2021. doi:10.1145/3446374

[29] I. J. Goodfellow et al., “Generative Adversarial Networks,” arXiv.org, https://arxiv.org/abs/1406.2661 (accessed Mar. 28, 2024).

[30] A. Creswell et al., “Generative Adversarial Networks: An overview,” arXiv.org, https://arxiv.org/abs/1710.07035 (accessed Mar. 28, 2024).

[31] M. Mirza and S. Osindero, “Conditional generative adversarial nets,” arXiv.org, https://arxiv.org/abs/1411.1784 (accessed Mar. 28, 2024).

[32] J. Ho, A. Jain, and P. Abbeel, “Denoising Diffusion Probabilistic models,” arXiv.org, https://arxiv.org/abs/2006.11239 (accessed Mar. 28, 2024).

[33] Y. Ren et al., “FastSpeech: Fast, robust and controllable text to speech,” arXiv.org, https://arxiv.org/abs/1905.09263 (accessed May 5, 2024).

[34] Y. Guo, C. Du, X. Chen, and K. Yu, “Emodiff: Intensity controllable emotional text-to-speech with soft-label guidance,” arXiv.org, https://arxiv.org/abs/2211.09496 (accessed May 5, 2024).

[35] R. Huang et al., “Prodiff: Progressive fast diffusion model for high-quality text-to-speech,” arXiv.org, https://arxiv.org/abs/2207.06389 (accessed May 5, 2024).

[36] S. Liu, D. Su, and D. Yu, “Diffgan-TTS: High-fidelity and efficient text-to-speech with Denoising Diffusion Gans,” arXiv.org, https://arxiv.org/abs/2201.11972 (accessed May 5, 2024).

[37] N. Chen et al., “Wavegrad: Estimating gradients for waveform generation,” arXiv.org, https://arxiv.org/abs/2009.00713 (accessed May 5, 2024).

[38] S. Lee et al., “Priorgrad: Improving conditional denoising diffusion models with data-dependent adaptive prior,” arXiv.org, https://arxiv.org/abs/2106.06406 (accessed May 5, 2024).

[39] O. Nazir and A. Malik, “Deep learning end to end speech synthesis: A Review,” *2021 2nd International Conference on Secure Cyber Computing and Communications (ICSCCC)*, May 2021. doi:10.1109/icsccc51823.2021.9478125

[40] Z. Khanjani, G. Watson, and V. P. Janeja, “Audio Deepfakes: A survey,” *Frontiers in Big Data*, vol. 5, Jan. 2023. doi:10.3389/fdata.2022.1001063

[41] M. Murphy, R. Metz, M. Bergen, and Bloomberg, “Biden Audio Deepfake Spurs AI startup Elevenlabs-valued at $1.1 billion-to ban account: ‘we’re going to see a lot more of this,’” Fortune, https://fortune.com/2024/01/27/ai-firm-elevenlabs-bans-account-for-biden-audio-deepfake/ (accessed May 5, 2024).

[42] M. Mcuba, A. Singh, R. A. Ikuesan, and H. Venter, “The effect of deep learning methods on Deepfake audio detection for digital investigation,” *Procedia Computer Science*, vol. 219, pp. 211–219, 2023. doi:10.1016/j.procs.2023.01.283

[43] J. Yi et al., “Audio deepfake detection: A survey,” arXiv.org, https://arxiv.org/abs/2308.14970 (accessed May 5, 2024).

[44] L. Cuccovillo et al., “Open challenges in synthetic speech detection,” arXiv.org, https://arxiv.org/abs/2209.07180 (accessed May 5, 2024).

[45] Y. Ning, S. He, Z. Wu, C. Xing, and L.-J. Zhang, “A review of deep learning based speech synthesis,” *Applied Sciences*, vol. 9, no. 19, Sep. 2019. doi:10.3390/app9194050

[46] N. Kanda et al., “Making flow-matching-based zero-shot text-to-speech laugh as you like,” arXiv.org, https://arxiv.org/abs/2402.07383 (accessed May 12, 2024).

[47] H. Tang, X. Zhang, J. Wang, N. Cheng, and J. Xiao, “Qi-TTS: Questioning intonation control for emotional speech synthesis,” *ICASSP 2023 - 2023 IEEE International*

Conference on Acoustics, Speech and Signal Processing (ICASSP), Jun. 2023. doi:10.1109/icassp49357.2023.10095623

[48] K. Zhou, B. Sisman, R. Rana, B. W. Schuller, and H. Li, "Speech synthesis with mixed emotions," *IEEE Transactions on Affective Computing*, vol. 14, no. 4, pp. 3120–3134, Oct. 2023. doi:10.1109/taffc.2022.3233324

[49] Z. Zhou, J. Chen, N. Wang, L. Li, and D. Wang, "Adversarial data augmentation for robust speaker verification," arXiv.org, https://arxiv.org/abs/2402.02699v1 (accessed May 5, 2024).

[50] J. Yamagishi, C. Veaux, S. King, and S. Renals, "Speech Synthesis Technologies for individuals with vocal disabilities: Voice banking and reconstruction," *Acoustical Science and Technology*, vol. 33, no. 1, pp. 1–5, 2012. doi:10.1250/ast.33.1

[51] S. Yang, M. Cheng, and S. Li, "Research and design of Intelligent Voice Customer Service System," *2023 IEEE 3rd International Conference on Information Technology, Big Data and Artificial Intelligence (ICIBA)*, May 2023. doi:10.1109/iciba56860.2023.10165037

[52] D. Xin et al., "Improving speech prosody of audiobook text-to-speech synthesis with acoustic and textual contexts," arXiv.org, https://arxiv.org/abs/2211.02336 (accessed May 5, 2024).

[53] Z. Zhang et al., "Speak foreign languages with your own voice: Cross-lingual neural codec language modeling," arXiv.org, https://arxiv.org/abs/2303.03926 (accessed May 5, 2024).