VILNIUS UNIVERSITY

FACULTY OF MATHEMATICS AND INFORMATICS

DATA SCIENCE STUDY PROGRAMME

Master's thesis

Data Selection Strategies for Multi-Speaker Text-to-Speech Synthesis in Lithuanian

Work Title in Lithuanian

Aleksandr Jan Smoliakov

Supervisor : Dr. Gerda Ana Melnik-Leroy

Scientific advisor : Dr. Gražina Korvel

Reviewer : pedagogical/scientific title Name Surname

Vilnius
2025

List of Figures

List of Tables

Contents

Introduction

The goal of creating machines that can speak like humans has captivated researchers for centuries. One of the earliest known attempts dates back to the 18th century, with Wolfgang von Kempelen's mechanical speech machine that utilized a bellows-driven lung and physical models of the tongue and lips.

Over the centuries, advancements in technology and understanding of human speech have driven significant progress in this field. Today's state-of-the-art systems, dominated by end-to-end (E2E) neural models, have achieved highly naturalistic speech with unprecedented acoustic quality. Notably, these end-to-end systems have unified the entire synthesis process into a single neural network, eliminating the need for complex multi-stage pipelines.

Training high-quality TTS models typically requires large amounts of annotated speech data. The common recommendation is to use at least 10 hours of recorded speech from a single speaker to achieve good results.

Liepa 2 is a recently released Lithuanian speech corpus that contains 1000 hours of annotated speech; however, this data is distributed across more than 2600 speakers, with most speakers contributing only a few minutes of speech. The top speaker has around 2.5 hours of recorded speech.

Training a high-quality single-speaker TTS model on such limited data poses a challenge. Multi-speaker TTS models can utilize data from multiple speakers to improve performance. However, training on all available data is a time-consuming and computationally expensive process, especially in the context of a master's thesis.

Therefore, it makes sense to explore strategies for selecting smaller subsets of the available data for training. The question that arises is, what is the best way to sample multi-speaker data for training TTS models?

This thesis aims to answer the following research questions:

- TODO

Scope: This study is exclusively focused on the Lithuanian language and the Liepa 2 speech corpus. It investigates a fixed total training data size of 30 hours. The models are limited to Tacotron 2 with DDC and FastPitch architectures within the Coqui TTS framework, using a pre-trained WaveGlow vocoder for waveform generation.

Limitations: The findings may not generalize to other languages, datasets with different characteristics, or other TTS architectures. The 30-hour training data size is a practical constraint and may not reflect performance at larger scales.

1 Literature review

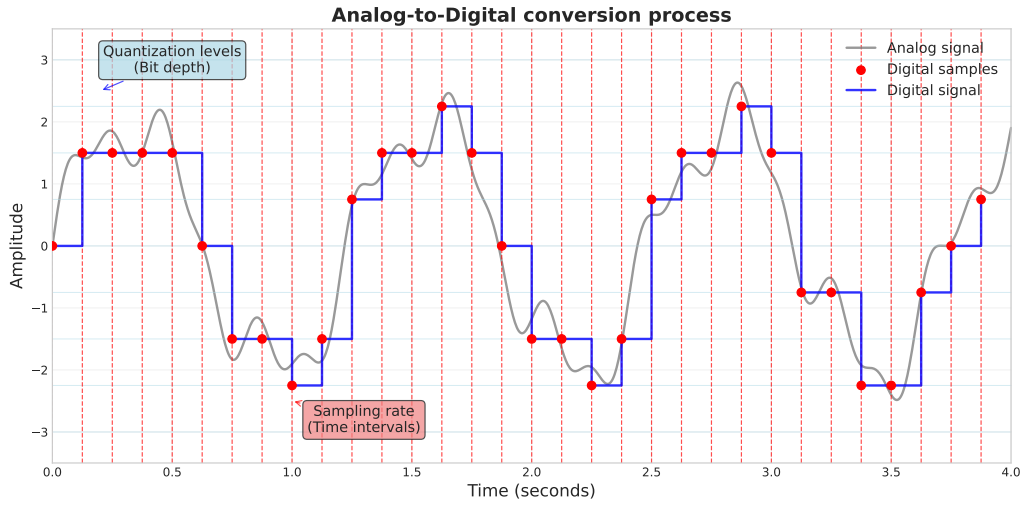
1.1 Digital representation of audio

Speech, or sound in general, is a continuous pressure wave that propagates through a medium, such as air. The key properties of sound waves include frequency (pitch), amplitude (loudness), and phase.

Converting continuous sound waves into a digital format suitable for computer processing involves two main steps: sampling and quantization.

Sampling is the process of measuring the amplitude of the sound wave at regular time intervals. The rate at which these samples are taken is called the sampling rate. According to the Nyquist-Shannon [1] sampling theorem, accurate reconstruction of a continuous signal requires a sampling rate that is strictly greater than twice the highest frequency present in the signal. The majority of human speech information is concentrated between 100 Hz and 8 kHz [2]. Thus, in text-to-speech applications, common sampling rates for audio are 22.05 kHz and 24 kHz, which can capture frequencies up to approx. 11 kHz and 12 kHz, respectively.

Quantization (also known as bit depth) is the mapping of continuous amplitude values to discrete levels for digital representation, which determines the precision of the representation. Common bit depths for audio are 16-bit and 24-bit formats. A visual representation of both sampling and quantization is provided in Figure 1.



***1** Visual representation of Analog-to-Digital conversion. The continuous grey line represents the analog signal. The vertical lines represent the **sampling rate** (time intervals), and the horizontal grid lines represent **quantization levels** (bit depth).*

Pre-emphasis is a high-frequency filtering technique applied to audio signals before further processing. Natural speech signals tend to have more energy in the lower frequencies, with a gradual drop-off towards higher frequencies (typically around -6 dB per octave). Pre-emphasis compensates for this spectral tilt by boosting high frequencies using a first-order high-pass filter, which is defined as:

$$y[n] = x[n] - \alpha x[n-1] \tag{1}$$

where $y[n]$ is the pre-emphasized signal, $x[n]$ is the original signal, α is the pre-emphasis coefficient (typically between 0.9 and 1.0, and often set to 0.97), and n is the sample index.

This transformation balances the frequency spectrum, improving the signal-to-noise ratio for higher frequencies and preventing the model from optimizing only for low-frequency components.

1.2 Time-Frequency Analysis

1.2.1 Fourier Transform

Fourier Transform (FT) is a mathematical technique that transforms a time-domain signal (such as an audio waveform) into its frequency-domain representation. The signal is decomposed into a sum of sine and cosine waves at various frequencies, each with a specific amplitude and phase. This allows us to analyze the frequency content of the signal.

Short-Time Fourier Transform [3] (STFT) extends the FT by applying it to short, overlapping segments (frames) of the signal. This transformation provides a time-frequency representation, showing how the frequency content of the signal changes over time.

In TTS applications, the STFT is computed by dividing the audio signal into short frames (usually, 20-50 ms) with a certain overlap (usually, 50-75%) between frames, windowed by a Hamming or Hann function to reduce the spectral leakage.

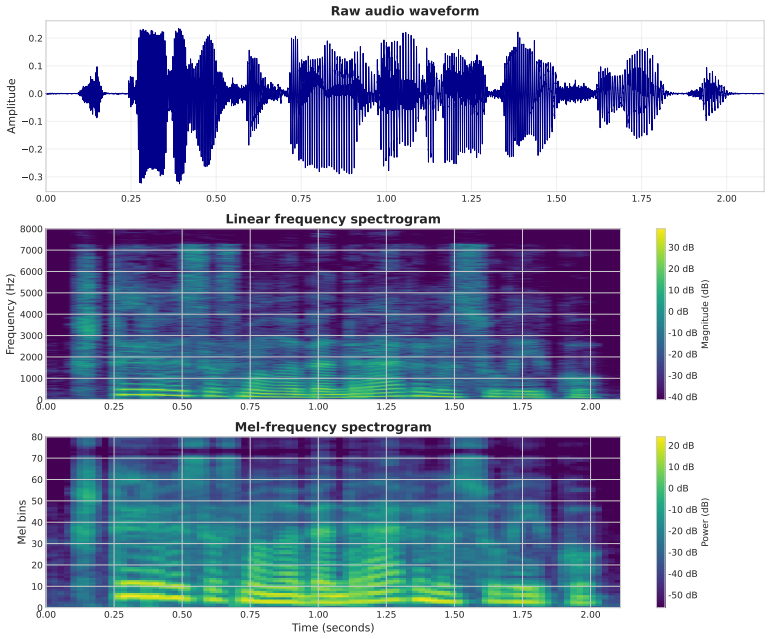
1.2.2 Spectrogram and Mel-spectrogram

The spectrogram is a visual representation of the STFT, displaying frequency on the vertical axis, time on the horizontal axis, and amplitude represented by the color intensity.

However, the human ear does not perceive frequencies linearly — it is more sensitive to lower frequencies than higher ones. To mimic this perceptual characteristic, the Mel scale [4] maps linear frequency f (in Hz) to a perceptual scale m (in Mels) using the following formula:

$$m = 2595 \cdot \log_{10}\left(1 + \frac{f}{700}\right) \tag{2}$$

Mel-spectrograms are computed by applying a Mel filterbank of overlapping triangular filters (or kernels) to the magnitude spectrogram obtained from the STFT. This results in a compressed representation of the audio signal that aligns more closely with human auditory perception. Such Mel-spectrograms are commonly used as input features for modern TTS systems. The differences between the raw waveform, the standard spectrogram, and the Mel-spectrogram are illustrated in Figure 2. Note how the Mel-spectrogram has a higher resolution in the lower frequencies, where the majority of the speech energy is concentrated.



2 *Raw audio waveform (top), its spectrogram (middle), and Mel-spectrogram (bottom) representations for the utterance "Štai ir visas mano bendravimas su vaiku".*

1.3 Text-to-speech synthesis

Text-to-Speech (TTS) synthesis, also known as speech synthesis, is the process of converting written text into human-like spoken words. Nowadays TTS is a key technology in numerous applications, including virtual assistants, accessibility tools, and language learning platforms.

1.3.1 Traditional TTS approaches

Early attempts at artificial speech synthesis evolved from the first mechanical devices in the 18th century to electronic systems. Wolfgang von Kempelen's mechanical speech machine demonstrated basic phoneme production using a physical model of the vocal tract. In 1937, Homer Dudley's invention of the Voder [5] became the first electronic speech synthesizer that could produce intelligible speech through operator-controlled acoustic parameters, establishing the foundation for modern electronic synthesis methods.

In the decades that followed, two main approaches for speech synthesis emerged: concatenative synthesis and parametric synthesis.

1.3.2 Concatenative synthesis

The concatenative synthesis approach [6] synthesizes speech by piecing together pre-recorded segments of human speech. This method involves several steps. First, it requires pre-recording a large database of speech segments spoken by a human voice actor in pristine, highly controlled studio conditions to ensure consistent audio quality and minimize background noise. Each segment is labeled and indexed based on its phonetic and prosodic properties.

During synthesis, the system breaks down the input text into short linguistic units (such as

phonemes or syllables) using a text analysis module. Then, it queries the speech database to find the best-matching segments for each unit using selection cost functions [7]. The retrieved segments are blended and concatenated to form a continuous speech waveform. Finally, the system uses signal processing techniques to smooth the transitions between segments and adjust pitch and duration to match the desired output characteristics.

Concatenative synthesis can produce natural-sounding individual speech units, but the final audio often has noticeable audible discontinuities at the concatenation points [7]. The segments may not blend smoothly due to differences in pitch, duration, and timbre. The prosody also tends to sound "choppy" and unnatural, since stringing disjointed segments together does not capture the natural rhythm and intonation patterns of connected speech.

Finally, concatenative synthesis requires language-specific expertise to design and maintain the underlying speech database and selection algorithms. This need for extensive data can make it challenging to develop concatenative TTS systems for low-resource languages or dialects.

1.3.3 Parametric synthesis

In contrast, statistical parametric speech synthesis [8] (SPSS) uses statistical models, typically Hidden Markov Models (HMMs) [9], to generate the parameters that control a speech waveform.

This method involves training a statistical model on a large corpus of recorded speech. The model learns the relationship between linguistic features (like phonemes and prosody) and the acoustic features of the speech signal, such as spectral envelope and fundamental frequency. During synthesis, the system takes text as input, converts it to a sequence of linguistic features, and then uses the trained model to generate a corresponding sequence of acoustic parameters.

Compared to concatenative synthesis, the statistical approach allows for more flexibility and control over the speech synthesis process, enabling the generation of a wider variety of voices and speaking styles. However, HMM-based synthesis [9] had a persistent problem: the statistical averaging built into the models tended to over-smooth the acoustic features, creating the characteristic "buzzy" or "muffled" sound that lacked the sharpness and detail of natural human speech.

1.4 Linguistic Representation (Text Processing)

In TTS systems, the input text must be pre-processed and converted into a suitable linguistic representation that the synthesis model can use. The main goal is to map the raw sentences into a sequence of symbols that can be more closely mapped to the acoustic features of speech.

Although theoretically an end-to-end TTS model could learn to map raw text directly to audio, in practice, pre-processing the text makes the model convergence easier and improves the quality of the synthesized speech.

This process typically involves several steps, such as text normalization, grapheme-to-phoneme conversion, and possibly prosody prediction.

1.4.1 Text normalization

Text normalization [10] is the process of converting raw written text into a more standardized "spoken" form. Typical steps include expanding abbreviations (e.g., expanding "Dr." to "Doctor"), punctuation removal, number normalization (e.g., converting "123" to "one hundred twenty-three"), and lowercasing.

As an example, the input text "Dr. Smith has 2 cats." could be normalized to "doctor smith has two cats".

Text normalization helps reduce the variability and complexity in the input text, decreases the number of unique symbols, and removes the ambiguities that could confuse the TTS model. The resulting normalized text is not only easier for the model to process, but can also be further converted into phonemes, which provide an even closer representation of the spoken language.

1.4.2 Graphemes vs. Phonemes

Text-to-speech systems use a discrete input representation derived from text, generally divided into grapheme-based or phoneme-based sequences.

Grapheme-based models ingest raw character sequences (orthography). This approach simplifies the inference pipeline by eliminating the dependency on external grapheme-to-phoneme (G2P) converters. However, it forces the model to implicitly learn pronunciation rules from data, which can be a significant challenge for languages with complex orthographies or inconsistent grapheme-to-phoneme mappings (e.g., "read" vs. "read").

In contrast, the phoneme-based approach uses a phonetic transcription of the text, typically in the International Phonetic Alphabet (IPA) or ARPABET form. By resolving pronunciation ambiguities prior to training, phonemes provide a more direct mapping to acoustic features, simplifying the model's task of learning alignment. The downside is that this approach requires an external grapheme-to-phoneme (G2P) conversion step [11]. Additionally, errors in the G2P conversion can propagate to the TTS model, affecting the quality of the synthesized speech.

There is another approach that augments the grapheme-based representation with explicit lexical stress markers or diacritics (e.g., tilde, acute, grave accents). This intermediate method helps the model disambiguate pronunciation of homographs and easier learn prosodic patterns without requiring a full phonetic transcription, particularly in languages where stress placement alters meaning.

1.4.3 Specific challenges in Lithuanian

Lithuanian is a Baltic language with a rich inflectional morphology and complex prosodic structure. It is a pitch-accent language with free stress, meaning the stress can fall on any syllable in a word, and can change the position depending on the grammatical form.

Challenges in Lithuanian TTS synthesis include:

- **High OOV rate:** Due to extensive word inflection, the number of unique word forms is signifi-

cantly higher than in English. This leads to data sparsity issues where many valid word forms may not appear in the training set.

- **Ambiguity without accentuation:** Typically, stress marks are omitted in written Lithuanian. However, stress position and tone (acute, circumflex, or short) determine the meaning of monographic words. Examples are shown in Table 1. A grapheme-based model without accentuation marks may struggle to predict the correct prosody, resulting in monotonic or mispronounced speech.

Word	Accentuation	Meaning
Antis	*ántis* (Acute) *añtis* (Circumflex)	A duck (noun) Bosom/Chest (noun)
Kasa	*kãsa* (Circumflex) *kasà* (Short)	He/she digs (verb) Braid/Pancreas (noun)

***1** Examples of Lithuanian homographs where accentuation determines meaning. A grapheme-only model cannot distinguish these without context or explicit stress marks.*

To overcome these challenges, tools like **Kirčiuoklis** (Vytautas Magnus University) are often employed in the text normalization pipeline. Kirčiuoklis automatically assigns stress marks to raw text. One weakness of Kirčiuoklis is that it relies on simple word-dictionary based lookup, which does not take into account the context of the word. Thus, it suggests multiple possible accentuation variants for homographs, leaving it up to the user to select the correct one.

In the absence of a high-quality, context-aware Grapheme-to-Phoneme (G2P) converter for Lithuanian, this thesis will focus on grapheme-based TTS synthesis with accentuation marks provided by Kirčiuoklis. In cases where Kirčiuoklis suggests multiple accentuation variants for a word, no stress marks will be added, leaving the TTS model to infer the correct prosody from context.

1.5 Embeddings and Representation Learning

1.5.1 The Concept of Embeddings

In machine learning, embeddings are dense vector representations of discrete entities (such as words, characters, or speakers) to a high-dimensional continuous vector space. Unlike one-hot encodings, which are sparse and highly dimensional, embeddings provide a dense, lower-dimensional representation that captures semantic relationships between underlying entities. For instance, in word embeddings, similar words tend to have more similar (correlated) vector representations, while dissimilar words map to more distant points in the vector space.

1.5.2 Text Embeddings

The “Encoder” part of a TTS model is responsible for converting a sequence of input symbols (characters or phonemes) into a sequence of feature vectors. Usually, this is done using an embed-

ding layer, which maps each “categorical” input symbol to a learnable fixed-size vector representation. During training, these embeddings are learned jointly with the rest of the TTS model.

1.5.3 Speaker Embeddings

To enable multi-speaker synthesis, TTS models require a representation of the speaker’s identity. There are several techniques for incorporating speaker information into TTS models:

Lookup Tables (LUT): Early multi-speaker approaches used simple, learnable embeddings where each speaker ID is mapped to a unique vector. The vectors are initialized randomly and learned jointly with the TTS model. While this method is straightforward and efficient, it cannot generalize to speakers not seen during training.

d-vectors and x-vectors: Transfer learning approaches [12] have demonstrated adapting speaker verification models for multispeaker TTS synthesis, enabling better speaker adaptation and higher voice quality. The general architecture of such a speaker encoder is illustrated in Figure 3. A speaker encoder model pre-trained on a massive, noisy dataset with thousands of speakers (e.g., the VoxCeleb dataset) learns the general speaker space. Its pre-trained weights are frozen and used to extract embeddings for the TTS training data, allowing the TTS model to effectively account for multi-speaker variation.

3 *General architecture of a Speaker Encoder. A reference audio of arbitrary length is processed (typically by LSTM or TDNN layers) and pooled to produce a fixed-length embedding vector (e.g., d-vector) representing the speaker identity.*

d-vectors: d-vectors [13] are fixed-length speaker embeddings derived from a separate speaker verification model. A reference encoder network takes a reference audio recording of arbitrary length and compresses it into a fixed-length vector known as a d-vector, that summarizes the speaker’s timbral and prosodic characteristics. These d-vectors are then provided as additional input to the TTS model, and are kept fixed during TTS training.

x-vectors: An evolution of d-vectors, x-vectors [14] use a Time Delay Neural Network (TDNN) architecture to capture the temporal context more effectively. These embeddings have shown an improved ability in zero-shot TTS scenarios.

One limitation of d-vectors and x-vectors is that if the reference audio is of poor quality or contains background noise, the resulting speaker embedding may not accurately represent the speaker’s identity, leading to degraded synthesis quality.

1.6 Deep learning for TTS

The limitations of complex, multi-stage pipelines motivated the creation of the end-to-end (E2E) model. E2E systems learn the entire speech synthesis process — from input text directly to acoustic output — using a single neural network. This approach promised to eliminate the need for hand-crafted pipelines that were difficult to design, required extensive expertise, and suffered from errors that accumulated across multiple components. By learning directly from text-audio pairs, E2E

models showed they could produce speech with higher naturalness and expressiveness than previous methods, representing a significant leap in TTS technology.

Although deep learning TTS models are more robust to variations in data quality compared to concatenative approaches, they are essentially "data-hungry" beasts that require large amounts of training data to achieve optimal performance. Their performance follows general scaling laws [15], improving as the amount of training data increases.

However, in the context of multi-speaker synthesis, there is a trade-off between the breadth of the data (number of distinct speakers) and the depth of the data (duration of audio per speaker). In theory, training on a dataset with a massive number of speakers, even with limited data per speaker, may allow the model to learn a more generalized latent space of voice characteristics. This high variance in the training data could act as a form of regularization, potentially improving the model's ability to generalize to unseen speakers (zero-shot capabilities) and adapt to new voices with minimal data (few-shot adaptation). In contrast, datasets with fewer speakers but high duration per speaker allow the model to capture fine-grained prosodic details specific to those voices, potentially achieving higher stability but lower generalization capabilities.

1.6.1 Feedforward neural networks

Feedforward Neural Networks (FNNs) are the simplest type of artificial neural networks, consisting of layers of interconnected nodes (neurons) where information flows in one direction — from the input, through hidden layers, to the output. While FNNs can be useful for basic regression or classification tasks, they lack the memory and context-awareness needed for processing sequential data like text and speech. Therefore, FNNs are not suitable for modelling TTS tasks that require understanding of temporal dependencies.

1.6.2 Encoder-Decoder architectures

The Encoder-Decoder architecture is a neural network architecture consisting of two components, namely an encoder and a decoder. The encoder processes the input data and compresses it into a high-dimensional latent representation. This vector captures the meaningful features of the input. The decoder uses this latent representation as context to generate the final output. This architecture is commonly used in sequence-to-sequence tasks, such as machine translation (text-to-text) and text-to-speech synthesis (text-to-audio frames).

1.6.3 Sequence-to-Sequence models and Tacotron 2

A common approach in modern neural TTS is the sequence-to-sequence (seq2seq) framework, which uses an encoder-decoder architecture with an attention mechanism to map input (text) sequences to output (audio) frames.

Tacotron [16] and Tacotron 2 [17] are two notable TTS models based on the sequence-to-sequence architecture. The variant that this thesis primarily focuses on is Tacotron 2 with Dynamic Convolutional Attention (DCA). The complete architecture of Tacotron 2 is depicted in Figure 4.

This architecture has three main components:

1. **Encoder:** The encoder's input is a character or phoneme sequence. A stack of convolutional layers followed by a bidirectional LSTM converts the character sequence into a high-level hidden feature representation.

2. **Attention mechanism:** A location-sensitive attention mechanism learns to align the high-level text representation with the decoder steps. This alignment helps the model determine which parts of the input text should be attended to when generating each of the output audio frames.

3. **Decoder:** Finally, the decoder is an autoregressive LSTM-based recurrent neural network that generates the Mel-spectrogram frame by frame.

While Tacotron 2 can generate high-quality speech, its autoregressive nature makes inference slow, as each audio frame must be generated sequentially. Additionally, the attention mechanism can sometimes fail, leading to issues like skipped or repeated words in the synthesized speech.

4 *The Tacotron 2 architecture. Note the recurrent connections in the decoder and the attention mechanism aligning encoder outputs to decoder steps. [17]*

1.6.4 Non-autoregressive models and FastPitch

To address the slow inference speed and stability issues of autoregressive models, non-autoregressive (parallel) models were developed. FastPitch [18] is a notable example of such a model that replaces the recurrent layers with Transformer architecture blocks relying on self-attention.

Unlike Tacotron 2, FastPitch model generates the entire Mel-spectrogram in parallel, significantly speeding up the inference. It utilizes a feed-forward Transformer encoder and decoder. A key component of FastPitch is the explicit modeling of prosody through pitch and duration predictors:

- **Duration predictor:** Since the input text length does not match the output audio length, FastPitch requires an external aligner (or unsupervised alignment learning) to train a duration predictor. This module predicts how many audio frames correspond to each input character.

- **Pitch predictor:** A separate module predicts the fundamental frequency (F_0) for every character. This pitch contour is projected and added to the latent representation before decoding.

The explicit duration modeling allows FastPitch to control the length of the generated speech and upsample the encoder output to match the target Mel-spectrogram length. The pitch predictor enables explicit control over intonation.

The high-level architecture of FastPitch is shown in Figure 5.

5 *The FastPitch architecture. It utilizes a feed-forward Transformer and explicit duration and pitch predictors, allowing for parallel generation of the Mel-spectrogram. [18]*

The primary advantages of FastPitch over Tacotron 2 are inference speed (due to non-autoregressive generation), robustness (no attention failures like skipping or repeating words), and controllability (pitch and speed can be tweaked manually during synthesis).

1.6.5 Other notable TTS models

Besides Tacotron 2 and FastPitch, other notable TTS architectures include **Glow-TTS**, which uses flow-based generative models for parallel inference, and **VITS** [19] (Conditional Variational Autoencoder with Adversarial Learning), which combines the acoustic TTS model (Glow-TTS) with a neural vocoder (HiFi-GAN) into a single end-to-end architecture.

1.6.6 Neural vocoders

Acoustic TTS models like Tacotron 2 and FastPitch generate intermediate acoustic features (Mel-spectrograms), but do not directly produce raw audio waveforms. Spectrograms are lossy representations that only capture the magnitude of the sound frequency bands, discarding phase information. Converting a lossy spectrogram into audio is a non-trivial task, as the phase information must be estimated. This challenge is known as the *inversion problem*.

The component used to convert spectrograms back into waveforms is called a vocoder.

Traditionally, the Griffin-Lim algorithm [20] has been used to iteratively estimate and reconstruct the phase information from the magnitude spectrogram. However, this method often produces audio with noticeable artifacts and lower quality compared to natural speech.

Modern TTS systems use neural vocoders, which are deep generative models trained to map acoustic features to raw waveforms. **WaveNet** [21] was one of the first autoregressive models to produce high-fidelity audio, but its sequential generation process made it prohibitively slow for real-time applications.

To address the speed limitations, Generative Adversarial Network (GAN) based vocoders were introduced. **HiFi-GAN** [22] is currently one of the state-of-the-art neural vocoders. It consists of a Generator that upsamples the Mel-spectrograms using transposed convolutions and a set of Discriminators (multi-scale and multi-period discriminators) that ensure the generated audio is indistinguishable from real human speech. HiFi-GANs are highly efficient and capable of faster-than-real-time synthesis on consumer hardware while maintaining high perceptual quality.

The framework used in this thesis, Coqui TTS [23], comes with a pre-trained HiFi-GAN v2 vocoder trained on a large multi-speaker dataset (VCTK [24]) with 110 English speakers. Given the HiFi-GAN v2's widespread adoption and status as a state-of-the-art neural vocoder, this thesis will use the HiFi-GAN v2 model for waveform generation. In order to ensure compatibility between the TTS models and the vocoder, the TTS models' acoustic parameters will be configured to the exact same settings (sampling rate, FFT parameters, Mel filterbank settings) as those used during the HiFi-GAN v2's training.

1.7 Multi-speaker TTS

Multi-speaker TTS models are designed to synthesize speech in the voices of multiple speakers. In order to achieve this, these models are indeed trained on data from many different speakers, allowing them to learn the characteristics of each voice and synthesize speech that sounds like a specific individual, while still being able to generalize the shared linguistic and acoustic patterns across speakers.

1.7.1 Techniques

In multi-speaker models, the network is conditioned on a speaker embedding. The model learns a shared representation of phonetics (how text maps to sound generally) while using an additional input — the speaker embedding — to adjust the timbre and prosodic characteristics specific to a voice.

Early successful implementations of this approach include Deep Voice 2 [25], which demonstrated effective multi-speaker synthesis by learning speaker-specific embeddings.

As mentioned in an earlier section, the speaker embeddings can be learned jointly with the TTS model during training, or they can be pre-trained using a separate speaker verification model.

1.7.2 Challenges

One key challenge in multi-speaker TTS is ensuring that the model can generalize across many speakers while still maintaining high quality for each. There is a trade-off between the *breadth* of the dataset (number of speakers) and the *depth* (minutes of audio per speaker).

Standard TTS systems historically required 10 to 20 hours of recorded speech for a single professional speaker. However, deep learning models capable of *transfer learning* can produce intelligible speech for a new speaker with significantly less data, potentially as little as a few minutes — if the base model has been pre-trained on a sufficiently diverse multi-speaker dataset.

1.8 Data selection in TTS

As TTS models require large amounts of annotated speech data for training, training on all available data can be computationally prohibitive and time-consuming.

In this thesis, the efficiency of such TTS models will be evaluated by fixing the total dataset size while varying the distribution:

- **High-resource per speaker:** Fewer speakers (30), but high fidelity (45 min each).
- **Balanced:** Moderate diversity (60 speakers), moderate data (22.5 min each).
- **High-diversity:** Many speakers (180), low resource (7.5 min each).

This experimental design aims to determine whether modern architectures benefit more from observing a wider variety of vocal characteristics (high diversity) or from observing more detailed

prosodic and acoustic patterns for specific voices (high resource), particularly when evaluated on a set speakers seen during training.

1.9 Evaluation metrics

Evaluating Text-to-Speech systems is notoriously difficult because “quality” is a subjective metric defined by human perception. There is no single mathematical objective function that perfectly correlates with human judgement of naturalness and intelligibility. Therefore, TTS systems are typically evaluated using subjective listening tests.

1.9.1 Mean Opinion Score (MOS)

The most standard metric for evaluating speech synthesis quality is the Mean Opinion Score (MOS), originally derived from telecommunications quality standards (ITU-T P.800). [26].

In a MOS test, human listeners (raters) are presented with a set of synthesized speech audio samples and asked to rate them on a 5-point Likert scale. The standard scale for “Naturalness” is:

- **5:** Excellent (Imperceptible difference from real speech)
- **4:** Good (Perceptible but not annoying)
- **3:** Fair (Slightly annoying)
- **2:** Poor (Annoying)
- **1:** Bad (Very annoying / Unintelligible)

The final score is the arithmetic mean of all ratings collected for a specific TTS system. Although MOS is subjective, with a sufficient number of raters (typically, at least 15-20), the scores tend to converge and provide a reliable ranking between different models.

1.9.2 Latin square design

A major challenge in subjective listening tests is controlling for biases. If a rater hears the same sentence produced by different TTS systems in a row, their ratings may be influenced by the repetition (repetition effect) or by the relative order of presentation (order effect). For instance, a “Slightly annoying” sample may be rated more harshly if it follows an “Excellent” sample (contrast effect).

In order to mitigate these biases, a Latin square design [27] is often employed for MOS tests. In this experimental design:

1. A set of test sentences (utterances) is selected.
2. The listeners are divided into groups.
3. The presentation is balanced such that each listener hears every test sentence exactly once, and every TTS system (model) exactly once per block of trials, but never the same sentence-system combination twice.

For a multi-speaker TTS evaluation (as is the case in this thesis), the Latin square design ensures that the ratings reflect the quality of the model rather than the linguistic content of the sentence or listener fatigue. By rotating the systems and sentences across listener groups, the influence of specific difficult sentences is averaged out across all models.

1.10 Summary

Modern TTS synthesis has evolved from complex, multi-stage concatenative and statistical pipelines to end-to-end deep learning architectures. Models such as Tacotron 2 and FastPitch, combined with neural vocoders like HiFi-GAN, are capable of generating natural-sounding speech that can be indistinguishable from human speech. These systems rely on digital signal processing techniques to convert raw audio into representations like Mel-spectrograms — and learned embeddings to handle text-to-audio alignment and speaker identity.

1.11 Research Gap

2 Methodology

The following chapters will detail the methodology used to train these models on Lithuanian data variants and the results of subjective evaluation on the target speaker set.

2.1 Research design

The independent variables in this study are:

- Data selection strategy: Different methods for selecting subsets of the training data.
- TTS model architecture: Comparing different TTS architectures.

The dependent variables in this study are:

- TTS model performance: Measured using objective metrics (TODO, decide) and subjective evaluations (MOS scores).

The controlled variables in this study are:

- Dataset: The Liepa 2 Lithuanian speech corpus.
- Training data size: The same total amount of training data used across all experiments.
- Training procedure: The same training hyperparameters and protocols applied across all experiments.
- Evaluation metrics: The same objective and subjective evaluation methods applied across all experiments.

2.2 Data and preprocessing

2.2.1 Liepa 2 dataset

The primary dataset for this research is the Liepa 2 corpus, a comprehensive collection of Lithuanian speech data. The corpus encompasses 1000 hours of short voice recordings from 2621 speakers, along with corresponding text transcriptions. The recordings span various speech styles and contexts, including read speech (audiobooks, news), TV and radio broadcasts, spontaneous speech.

More specifically, the dataset contains:

- Total duration: 1000 hours of speech data.
- Number of speakers: 2621 unique speakers.
- Number of utterances: 1,874,648 recorded utterances.

2.2.2 Acoustic preprocessing

The raw audio recordings from the Liepa 2 dataset are sampled at 16 kHz. To prepare the audio for TTS model training, the audio waveforms are resampled to 22,050 Hz. While resampling to a higher frequency does not add new information, it will be compatible with pre-trained vocoders that expect 22,050 Hz input.

Additional preprocessing steps, such as silence trimming and normalization, are performed by the Coqui TTS framework during training.

The acoustic processing that transforms audio waveforms into Mel-spectrograms is performed on-the-fly during model training by the Coqui TTS framework. The Mel-spectrogram parameters used are as follows:

TODO move to appendix

- fft_size: 1024
- win_length: 1024
- hop_length: 256
- frame_length_ms: null
- frame_shift_ms: null
- stft_pad_mode: ”reflect”
- sample_rate: 22050
- resample: false
- preemphasis: 0.98

- ref_level_db: 20
- do_sound_norm: false
- log_func: ”np.log10”
- do_trim_silence: true
- trim_db: 60
- do_rms_norm: false
- db_level: null
- power: 1.5
- griffin_lim_iters: 60
- num_mels: 80
- mel_fmin: 0.0
- mel_fmax: 8000.0
- spec_gain: 20
- do_amp_to_db_linear: true
- do_amp_to_db_mel: true
- pitch_fmax: 640.0
- pitch_fmin: 1.0
- signal_norm: true
- min_level_db: -100
- symmetric_norm: true
- max_norm: 4.0
- clip_norm: true
- stats_path: null

2.2.3 Text normalization

Before feeding text data into TTS models, the text undergoes normalization to convert it into a more consistent and model-friendly format. The Liepa 2 text already includes a significant level of normalization - for instance, the following elements are written exactly as they were spoken: dates, times, acronyms, abbreviations, numbers.

However, some additional normalization steps are applied to further standardize the text:

- Punctuation standardization: Replacing uncommon punctuation marks with more common equivalents (e.g., replacing em dashes with hyphens, and semicolons with commas).
- Removal of extraneous characters: Eliminating any remaining characters that are not letters, digits, whitespace, or basic punctuation (.,-?!).
- Whitespace normalization: Collapsing multiple consecutive whitespace characters into a single space and trimming leading/trailing whitespace.
- Accent addition: Adding accent marks (tilde, acute, grave) to words using Kirčiuoklis tool (where Kirčiuoklis suggests multiple options, accent is not added).
- Lowercasing: Converting all text to lowercase to reduce vocabulary size.
- Letter replacements: Substituting non-Lithuanian letters with their Lithuanian equivalents ('w' with 'v', 'q' with 'kv', and 'x' with 'ks').

In order to add the accents to the text, the online tool Kirčiuoklis was queried with all unique words from the dataset. The tool returned the accented versions of the words, which were then used to replace the unaccented words in the text. In cases where Kirčiuoklis provided multiple accentuation options for a word, no accent was added to avoid introducing errors.

As a result of these normalization steps, the vocabulary size is reduced from 140 characters to 41 characters, simplifying the learning task for TTS models.

2.2.4 Speaker embeddings

For multispeaker TTS training, speaker embeddings are computed using a pre-trained speaker encoder model. The speaker embedding computation process involves:

- **Speaker encoder model**: Pre-trained model from Coqui TTS based on TODO
- **Embedding extraction**: Each speaker's audio samples are processed through the speaker encoder to generate 512-dimensional speaker embeddings (d-vectors).
- **Pooling strategy**: TODO
- **Normalization**: TODO

These speaker embeddings are then used during TTS training to condition the model on speaker identity, enabling the synthesis of speech in different voices.

2.3 Data selection strategies

To investigate the impact of data selection on TTS performance, several strategies for selecting subsets of the Liepa 2 dataset were employed. All strategies maintain a fixed total training budget of 30 hours to ensure fair comparison across experiments.

2.3.1 Speaker filtering criteria

Before applying selection strategies, the dataset undergoes initial filtering:

- **Speech type filtering**: Only "read speech" samples are used, excluding spontaneous speech to ensure consistent quality and pronunciation.
- **Age group filtering**: Speakers from age groups 18–25, 26–60, and 60+ are included, excluding children (0–17) to focus on adult speech patterns.
- **Gender balancing**: Equal representation of male and female speakers when possible.

2.3.2 Selection strategies

Top-N speaker (Speaker depth): This strategy prioritizes speaker depth by selecting the N speakers with the most available data. For multispeaker training with 20 speakers, the top 10 male and top 10 female speakers (by sample count) are selected. This approach maximizes the amount of data per speaker, potentially allowing models to learn more stable speaker-specific characteristics and alignment patterns.

Balanced speaker (Speaker breadth): This strategy prioritizes speaker diversity by distributing the training budget across a larger number of speakers, with each speaker contributing roughly equal amounts of data. The selection process aims to include as many speakers as possible while maintaining the 30-hour budget constraint and gender balance.

Random sampling: As a baseline, this strategy randomly samples utterances from the filtered dataset without regard to speaker distribution. This provides a control condition to assess whether structured speaker selection provides benefits over random data selection.

2.4 Experimental design

2.4.1 Model architectures

Three TTS model architectures are evaluated in this study:

Tacotron 2 with Double Decoder Consistency (DDC): The primary autoregressive model used in this study. DDC is an enhanced version of Tacotron 2 that employs dual decoders with different reduction factors (r=1 for the main decoder, r=7 for the coarse decoder) to improve attention alignment stability. This architecture is particularly suitable for multispeaker scenarios where attention alignment can be challenging.

FastPitch: A non-autoregressive model that predicts duration and pitch explicitly, allowing for parallel generation of Mel-spectrograms. FastPitch uses Transformer architecture instead of RNNs and should theoretically be less sensitive to attention alignment issues.

2.4.2 Training procedure

All models are trained using the Coqui TTS framework with consistent training procedures:

- **Training duration**: 400 epochs maximum with early stopping based on validation loss.
- **Optimization**: RAdam optimizer with learning rate scheduling using MultiStepLR (milestones at 10k, 20k, 30k, 40k steps with gamma=0.32).
- **Loss function**: Combination of decoder loss (α=0.25), post-net loss (α=0.25), SSIM losses (α=0.25), guided attention loss (α=5.0), and stop token loss (weight=15.0).
- **Batch size**: 64 for Tacotron 2 variants, 32 for FastPitch.
- **Gradient clipping**: 0.05 to prevent gradient explosion.
- **Validation**: 1% of training data held out for validation.

2.5 Model training configurations

2.5.1 Tacotron 2 with DDC hyperparameters

2 *Tacotron 2 DDC training configuration*

Parameter	Value
Batch size	64
Learning rate	0.001
Optimizer	RAdam
LR scheduler	MultiStepLR
Max epochs	400
Decoder reduction factor	1
DDC reduction factor	7
Attention type	Location-sensitive
Memory size	-1 (disabled)
Speaker embedding dim	512
Number of speakers	20
Stopnet	Enabled
Separate stopnet	True

***3** FastPitch training configuration*

Parameter	Value
Batch size	32
Eval batch size	16
Learning rate	0.001
Optimizer	RAdam
LR scheduler	MultiStepLR
Max epochs	400
Duration predictor layers	2
Pitch predictor layers	2
Transformer encoder layers	6
Transformer decoder layers	6
Attention heads	1
Encoder hidden dim	384
Decoder hidden dim	384

2.5.2 FastPitch hyperparameters

2.5.3 Vocoder configuration

For Mel-spectrogram to waveform conversion, a pre-trained HiFi-GAN v2 vocoder is used across all experiments. The vocoder was pre-trained on the VCTK corpus [24] and is compatible with the 22.05 kHz sampling rate used in this study. Using the same vocoder across all experiments ensures that differences in audio quality can be attributed to the TTS model rather than the vocoder.

2.6 Evaluation methodology

2.6.1 Objective evaluation metrics

Objective evaluation is performed using standard acoustic metrics:

- **Training convergence metrics:** Training and validation loss curves, attention alignment visualization, and convergence time.

2.6.2 Subjective evaluation

Subjective evaluation is conducted through a web-based listening test application developed specifically for this study. The evaluation protocol includes:

- **Rating scale:** 5-point Mean Opinion Score (MOS) scale where 1=Very Poor, 2=Poor, 3=Fair, 4=Good, 5=Excellent.

2.6.3 Computational hardware

The experiments were conducted on a personal high-performance computing setup with the following specifications:

- CPU: AMD Epyc 7642 48-Core, 96 thread processor
- RAM: 256 GB DDR4 3200 MHz
- GPU: NVIDIA GeForce RTX 3090 with 24 GB VRAM
- Storage: 2 TB NVMe SSD

2.6.4 Implementation framework

All experiments are implemented using the Coqui TTS framework, an open-source toolkit for training TTS models. The experimental pipeline is automated using Make build system with the following components:

- **Data preprocessing:** Automated scripts for audio conversion, text normalization, and metadata generation.
- **Speaker embedding computation:** Batch processing of speaker embeddings using pre-trained encoder models.
- **Training orchestration:** Automated model training with hyperparameter configuration and checkpointing.
- **Inference pipeline:** Batch synthesis of test sentences for evaluation purposes.
- **Evaluation tools:** Integration with subjective evaluation web application and objective metric computation.

2.7 Evaluation protocol

2.7.1 Test set

Evaluation was performed on a held-out test set of 100 phrases from the Liepa 2 dataset.

2.7.2 Objective evaluation

2.7.3 Subjective evaluation

Subjective evaluation was conducted using Mean Opinion Score (MOS) tests.

3 Results and analysis

3.1 Tacotron 2 with DDC

3.1.1 Objective results

3.1.2 Subjective results

3.1.3 Qualitative analysis

3.2 FastPitch

3.2.1 Objective results

3.2.2 Subjective results

3.2.3 Qualitative analysis

4 Conclusion

4.1 Summary of findings

4.2 Contributions

4.3 Limitations of the study

4.4 Future work

5 References

[1] C. E. Shannon. “Communication in the presence of noise.” In: *Proceedings of the IRE* 37.1 (1949), pages 10–21.

[2] L. R. Rabiner, R. W. Schafer. *Theory and applications of digital speech processing*. Pearson, 2010.

[3] A. V. Oppenheim, R. W. Schafer, J. R. Buck. *Discrete-time signal processing*. 2nd. Upper Saddle River, NJ: Prentice Hall, 1999.

[4] S. S. Stevens, J. Volkmann, E. B. Newman. “A scale for the measurement of the psychological magnitude pitch.” In: *The Journal of the Acoustical Society of America* 8.3 (1937), pages 185–190.

[5] D. H. Klatt. “Review of text-to-speech conversion for English.” In: *The Journal of the Acoustical Society of America* 82.3 (1987), pages 737–793.

[6] A. J. Hunt, A. W. Black. “Unit selection in a concatenative speech synthesis system using a large speech database.” In: *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*. Volume 1. IEEE. 1996, pages 373–376.

[7] A. W. Black, N. Campbell. “Optimising selection of units from speech databases for concatenative synthesis.” In: *Eurospeech*. 1995.

[8] H. Zen, K. Tokuda, A. W. Black. “Statistical parametric speech synthesis.” In: *Speech communication* 51.11 (2009), pages 1039–1064.

[9] K. Tokuda, Y. Nankaku, T. Toda, H. Zen, J. Yamagishi, K. Oura. “Speech synthesis based on hidden Markov models.” In: *Proceedings of the IEEE* 101.5 (2013), pages 1234–1252.

[10] R. Sproat, A. W. Black, S. Chen, S. Kumar, M. Ostendorf, C. Richards. “Normalization of non-standard words.” In: *Computer Speech & Language* 15.3 (2001), pages 287–333.

[11] M. Bisani, H. Ney. “Joint-sequence models for grapheme-to-phoneme conversion.” In: *Speech Communication* 50.5 (2008), pages 434–451.

[12] Y. Jia, Y. Zhang, R. Weiss, Q. Wang, et al. “Transfer learning from speaker verification to multispeaker text-to-speech synthesis.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. Volume 31. 2018.

[13] E. Variani, X. Lei, E. McDermott, I. L. Moreno, J. Gonzalez-Dominguez. “Deep neural networks for small footprint text-dependent speaker verification.” In: *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2014, pages 4052–4056.

[14] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, S. Khudanpur. “X-vectors: Robust DNN Embeddings for Speaker Recognition.” In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2018, pages 5329–5333.

[15] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, et al. “Scaling Laws for Neural Language Models.” In: *arXiv preprint arXiv:2001.08361* (2020).

[16] Y. Wang, R. Skerry-Ryan, D. Stanton, Y. Wu, et al. “Tacotron: Towards End-to-End Speech Synthesis.” In: *Interspeech*. 2017, pages 4006–4010.

[17] J. Shen, R. Pang, R. J. Weiss, M. Schuster, et al. “Natural TTS Synthesis by Conditioning Wavenet on MEL Spectrogram Predictions.” In: *International Conference on Machine Learning* (2018), pages 4779–4788.

[18] A. Łańcucki. “FastPitch: Parallel Text-to-speech with Pitch Prediction.” In: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2021, pages 6588–6592.

[19] J. Kim, J. Kong, J. Son. “Conditional Variational Autoencoder with Adversarial Learning for End-to-End Text-to-Speech.” In: *arXiv preprint arXiv:2106.06103* (2021).

[20] D. Griffin, J. Lim. “Signal estimation from modified short-time Fourier transform.” In: *IEEE Transactions on acoustics, speech, and signal processing* 32.2 (1984), pages 236–243.

[21] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, K. Kavukcuoglu. “Wavenet: A generative model for raw audio.” In: *arXiv preprint arXiv:1609.03499* (2016).

[22] J. Kong, J. Kim, J. Bae. “HiFi-GAN: Generative Adversarial Networks for Efficient and High Fidelity Speech Synthesis.” In: *NeurIPS*. 2020.

[23] Coqui. *Coqui TTS*. https://github.com/coqui-ai/TTS. 2021.

[24] J. Yamagishi, C. Veaux, K. MacDonald. *CSTR VCTK Corpus: English Multi-speaker Corpus for CSTR Voice Cloning Toolkit (version 0.92)*. https://datashare.ed.ac.uk/handle/10283/3443. doi:10.7488/ds/2645. 2019.

[25] A. Gibiansky, S. Arik, G. Diamos, J. Miller, K. Peng, W. Ping, J. Raiman, Y. Zhou. “Deep Voice 2: Multi-Speaker Neural Text-to-Speech.” In: *arXiv preprint arXiv:1705.08947* (2017).

[26] ITU-T. *P.800 : Methods for subjective determination of transmission quality*. Recommendation P.800. International Telecommunication Union, 1996.

[27] E. J. Williams. “Experimental Designs Balanced for the Estimation of Residual Effects of Treatments.” In: *Australian Journal of Chemistry* 2 (1949), pages 149–168. URL: https://api.semanticscholar.org/CorpusID:96306494.