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 [1] 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

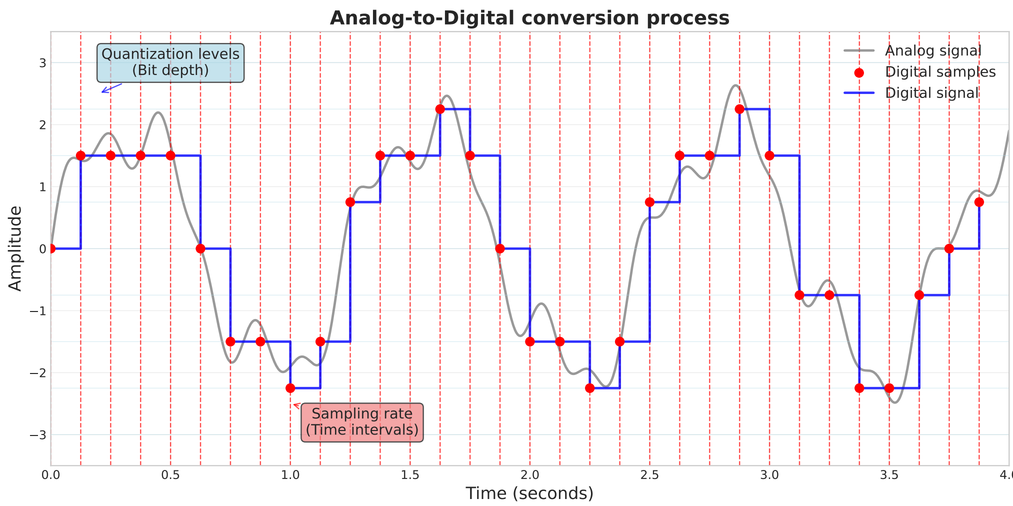
## 1.1 Digital representation of audio

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 [2] 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. Frequencies in the range between 300 Hz and 3400 Hz contribute most to human speech intelligibility and recognition. [3]. 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, $\alpha$ 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 [4] (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.

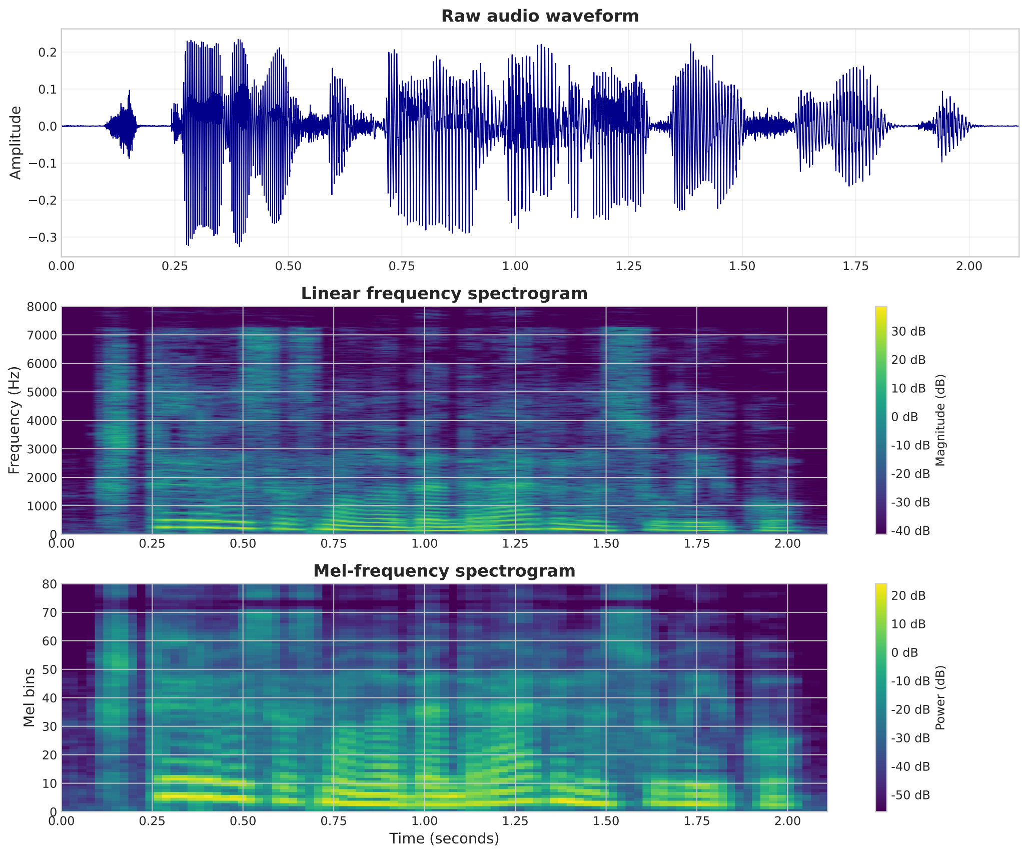
### 1.2.2 Spectrogram and Mel-spectrogram

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 [5] 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 1939, Homer Dudley's invention of the Voder [6] 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 [7] 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 [8]. 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 continuity distortions at the concatenation points [8]. 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 [9] (SPSS) uses statistical models, typically Hidden Markov Models (HMMs) [10], 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 [10] 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 [11] is the process of converting raw written text with non-standard words (NSWs) 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 [12]. 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 significantly 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 with accentuation marks has been shown to improve synthesis quality in Lithuanian. [13].

| Word | Accentuation | Meaning |
|---|---|---|
| Antis | *ántis* (Acute)<br>*añtis* (Circumflex) | A duck (noun)<br>Bosom/Chest (noun) |
| Kasa | *kãsa* (Circumflex)<br>*kasà* (Short) | He/she digs (verb)<br>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** [14] (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. [15]

### 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 embedding 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.

## 1.6 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” systems that require large amounts of training data to achieve optimal performance. Extrapolating from results in language modeling, it is observed that model performance follows general scaling laws [16], 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, preventing overfitting to noise and idiosyncrasies of individual speakers. 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 [17], which uses an encoder-decoder architecture with an attention mechanism to map input (text) sequences to output (audio) frames.

Tacotron [18] and Tacotron 2 [19] 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 3.

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 [20] mechanism learns to align the high-level text representation with the decoder steps. This alignment determines which parts of the input text should be attended to when generating each of the output audio frames.

3. **Decoder and Post-net:** The autoregressive LSTM decoder generates a coarse Mel-spectrogram frame. This output is then passed through a convolutional **Post-net** which predicts a residual to refine the spectral details and improve reconstruction quality.

4. **Stopnet:** A linear layer projects the LSTM decoder's output to a scalar, predicting the probability that the current frame is the "stop token", which halts the synthesis process. This allows the model to dynamically determine the output duration.

The model is optimized by minimizing a combination of losses: the mean squared error (MSE) between the predicted and ground truth Mel-spectrograms (both the Decoder and Post-net outputs), the spectral similarity index (SSIM) loss for improving spectral similarity, the "guided attention" loss to encourage diagonal attention alignments, and the binary cross-entropy loss for the stop token prediction.

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.

***3*** *The Tacotron 2 architecture. Note the recurrent connections in the decoder and the attention mechanism aligning encoder outputs to decoder steps. [19]*

### 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 [21] is a notable example of such a model that replaces the recurrent layers with Transformer [22] 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 predictor 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 4.

***4** The FastPitch architecture. It utilizes a feed-forward Transformer and explicit duration and pitch predictors, allowing for parallel generation of the Mel-spectrogram. [21]*

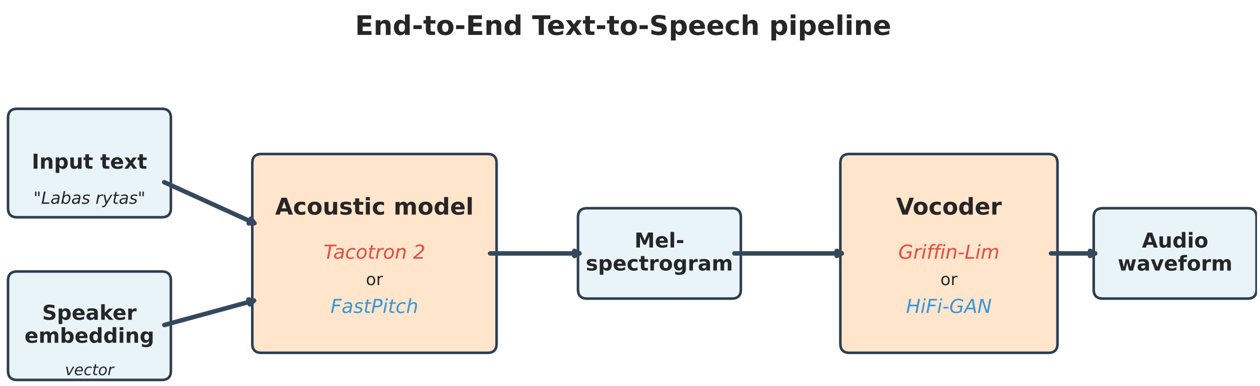
The primary advantages of FastPitch over Tacotron 2 are inference speed (due to non-autoregressive generation), robustness (no attention failures like skipping or repeating words), and controllability (pitch and speed can be tweaked manually during synthesis).

### 1.6.5 Other notable TTS models

Besides Tacotron 2 and FastPitch, other notable TTS architectures include **Glow-TTS** [23], which uses flow-based generative models for parallel inference, and **VITS** [24] (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

As illustrated in Figure 5, modern TTS systems typically employ a two-stage pipeline: an acoustic model 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*.




***5*** *Text-to-Speech synthesis pipeline. The TTS model generates Mel-spectrograms from input text, which are then converted to raw audio waveforms by a neural vocoder.*

An additional component called a vocoder is required to reconstruct the raw waveform from the Mel-spectrogram.

Traditionally, the Griffin-Lim algorithm [25] 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** [26] 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** [27] 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 [28], comes with a pre-trained HiFi-GAN v2 vocoder trained on a large multi-speaker dataset (VCTK [29]) with 110 English speakers.

The use of a vocoder trained on English data for Lithuanian synthesis is justified by the language-agnostic nature of the phase reconstruction task. Neural vocoders' primary function is to model the physics of human speech production rather than linguistic features. While language-dependent phonetic nuances exist, studies have shown that vocoders trained on large, diverse datasets can effectively generalize to unseen speakers and languages [30]. Therefore, this thesis will utilize the pre-trained HiFi-GAN v2 model for waveform generation.

This should allow the vocoder model to effectively generalize to unseen speakers and languages, provided that the acoustic feature parameters (including sampling rate, FFT size, Mel-filterbank limits) of the input Mel-spectrograms match those used during the vocoder's training.

Therefore, this thesis will utilize the pre-trained HiFi-GAN v2 model for waveform generation.

The TTS models' acoustic parameters will be configured to the exact same acoustic parameters used during the vocoder's training to ensure compatibility.

## 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 Speaker embeddings

To enable multi-speaker synthesis, TTS models require a representation of the speaker's identity. 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 [31], which demonstrated effective multi-speaker synthesis by learning speaker-specific embeddings.

Nowadays there are several techniques for incorporating speaker embeddings 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 [32] 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 6. A speaker encoder model pre-trained on a massive, noisy dataset with thousands of speakers (e.g., the VoxCeleb dataset [33]) 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.

***6** 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 [34] 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 [35] 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.

### 1.7.2 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 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.

### 1.8.1 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). [36].

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)

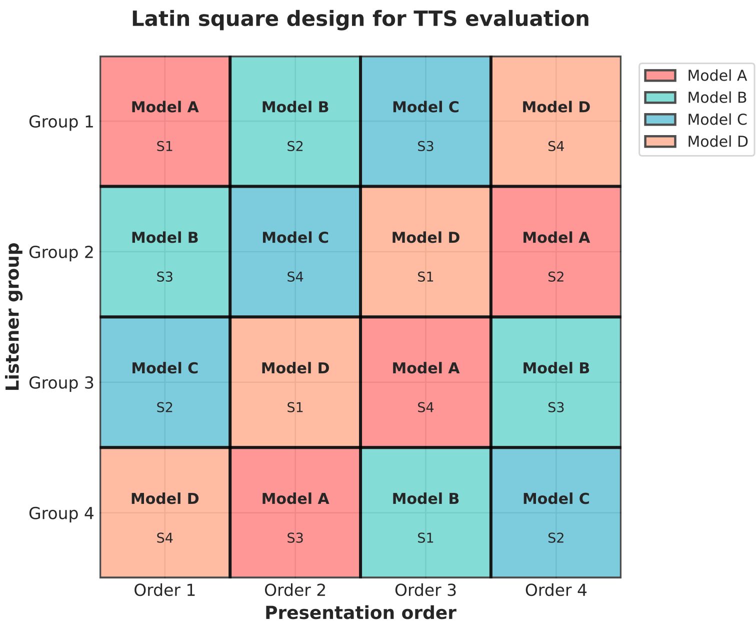
The final score is the arithmetic mean of all ratings collected for a specific TTS system. Although MOS is subjective, with a sufficient number of raters (typically, at least 15-20), the scores tend to converge and provide a reliable ranking between different models.

### 1.8.2 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 [37] 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.




*7 Latin square design for TTS evaluation. Each listener group hears each sentence exactly once, and each TTS system exactly once per block, ensuring balanced exposure and mitigating order/repetition biases.*

An example Latin square design for 4 TTS systems and 4 test sentences is illustrated in Figure 7.

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.

## 1.9 Research gap

While the literature demonstrates the capabilities of modern deep learning TTS architectures like Tacotron 2 and FastPitch to produce highly natural-sounding speech, several questions remain unanswered regarding their application to low-resource, morphologically complex languages like Lithuanian.

Firstly, although neural TTS models may follow general neural model scaling laws [16], implying that performance improves with more data, there is limited understanding of the optimal composition of training data under a fixed budget. In low-resource settings, scaling up the dataset size is not always feasible, and this may be further constrained by the computational resources required for training large models. A critical question is whether it is more beneficial to train on a smaller number of speakers with more data per speaker (high depth) or a larger number of speakers with less data per speaker (high breadth).

Current research primarily focuses on high-resource languages like English, where the availability of large, balanced multi-speaker datasets masks the nuances of this trade-off. For a pitch-accent language like Lithuanian, the requirements may be different. It is hypothesized that high-diversity datasets may help the model learn a richer representation of prosodic patterns, while high-depth datasets may improve the model's naturalness for the target speakers.

Secondly, most multi-speaker TTS research assumes access to large-scale datasets with thousands of utterances per speaker. There is a lack of research exploring how different TTS architectures (autoregressive vs. non-autoregressive) perform when the data per speaker is scarce (e.g., under 10 minutes).

This thesis aims to fill the research gap by systematically evaluating the efficiency of Tacotron 2 and FastPitch models trained on Lithuanian speech data. By controlling the total dataset size and varying the distribution of speakers and data per speaker, this study will provide insights into the optimal data composition for training multi-speaker TTS models in low-resource settings.

To summarize, the key research questions this thesis seeks to answer are:

- How does the trade-off between data breadth (number of speakers) and data depth (minutes per speaker) affect the performance of multi-speaker TTS models for Lithuanian?
- How do different TTS architectures (Tacotron 2 vs. FastPitch) perform under varying data selection strategies in low-resource settings?

The experiments will involve training models on three distinct data selection strategies:

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

The extreme low-depth condition (7.5 minutes per speaker) might pose convergence challenges for the models, especially for Tacotron 2, which relies on learning robust attention alignment. Thus, alignment convergence and training stability will be monitored to assess how data composition affects model robustness.

## 1.10 Summary

This literature review has provided an overview of the theoretical foundations required for modern Text-to-Speech synthesis. The evolution of TTS systems from mechanical apparatuses, through concatenative and statistical methods, to end-to-end deep learning architectures capable of generating natural-sounding speech has been discussed.

We have reviewed the entire TTS pipeline — from signal processing (sampling, quantization, Fourier transforms, and Mel-spectrogram extraction), through text normalization and representation (graphemes vs. phonemes), to deep learning architectures for acoustic modeling and vocoding. The literature highlights two architectures for acoustic modeling: the autoregressive Tacotron 2, known for high-quality spectral output but slow inference and stability issues, and the non-autoregressive FastPitch, which offers parallel generation with explicit control over pitch and duration.

We have examined the challenges specific to Lithuanian TTS synthesis. Unlike English, Lithuanian languages's high inflectional morphology leads to a large number of unique word forms, and its prosodic system requires handling of free stress and pitch accents, necessitating the use of tools like Kirčiuoklis for accentuation marking.

Finally, we have reviewed the role of speaker embeddings in enabling multi-speaker synthesis and the use of neural vocoders, specifically HiFi-GAN, to reconstruct high-fidelity waveforms from Mel-spectrograms. Despite these advancements, a gap remains in understanding how data diversity versus quantity affects model performance for complex, low-resource languages — a challenge this thesis addresses through the experiments detailed in the following chapters.

# 2 Methodology

This chapter details the experimental setup, data processing pipeline, training configurations, and evaluation protocol used to address the research questions regarding the optimal composition of multi-speaker training data for Lithuanian Text-to-Speech synthesis. The experiments are designed to systematically compare the performance of autoregressive and non-autoregressive models under varying degrees of data breadth and depth, while controlling for the total training budget.

## 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 foundation of this study is the **Liepa 2** Lithuanian speech corpus [1]. The corpus contains over 1000 hours of recorded speech 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.

Given the constraints of a fixed training budget and the computational cost of training large TTS models, a maximum total training time of 22.5 hours was established for all experimental models. This fixed size ensures that performance differences are attributable only to the distribution of the data (breadth vs. depth) and not the overall volume.

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 Data selection and speaker criteria

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.

#### 2.2.3 Speaker filtering criteria

The Liepa 2 corpus presents a challenge as most speakers contribute only a few minutes of audio. To ensure a valid comparison across data strategies, speakers were selected based on the following criteria:

1. **Minimum Duration:** Only speakers with at least 45 minutes of recorded speech were considered for the High-Depth condition. For other conditions, speakers were selected until the required per-speaker duration was met.
2. **Gender Balance:** The selection aimed to maintain a roughly equal split between male and female speakers to ensure the models generalize across pitch ranges.
3. **Speech type filtering**: Only "read speech" samples are used, excluding spontaneous speech to ensure consistent quality and pronunciation.
4. **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.
5. **Gender balancing**: Equal representation of male and female speakers when possible.

### 2.3 Experimental Design: Data Subsets

To investigate the trade-off between speaker diversity (breadth) and data quantity per speaker (depth), three distinct training datasets were constructed. To ensure a fair comparison, the **Total Dataset Size** was fixed at approximately 22.5 hours for all three experiments.

The configurations are defined as follows:

#### 2.3.1 Text Normalization and Accentuation

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.

**2** *Summary of the three experimental data subsets. The total duration is held constant while varying the number of speakers and the duration per speaker.*

| Subset Name | Strategy | Speakers ($N$) | Time/Speaker | Total Time |
|---|---|---|---|---|
| **Set-Depth** | High Fidelity | 30 | 45 min | 22.5 hours |
| **Set-Balance** | Balanced | 60 | 22.5 min | 22.5 hours |
| **Set-Breadth** | High Diversity | 180 | 7.5 min | 22.5 hours |

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

The text processing pipeline was configured to handle the specific challenges of the Lithuanian language's prosody:

1. **Grapheme Input:** The models were trained using a grapheme-based representation, augmented with accentuation marks.
2. **Cleaning:** Non-standard characters present in the Liepa 2 transcripts were removed.
3. **Normalization:** Numbers, abbreviations, and symbols were expanded into full words (e.g., "2023 m." $\rightarrow$ "du tūkstančiai dvidešimt tretieji metai").
4. **Accentuation:** The raw text was processed using **Kirčiuoklis** [14] for automatic stress assignment. Consistent with the literature review, in cases where Kirčiuoklis suggested multiple stress possibilities for a homograph, the text was left unaccentuated, forcing the TTS model to infer the correct prosody from the sentence context.

### 2.3.2 Audio Preprocessing

All audio data was uniformly preprocessed to ensure compatibility with the pre-trained neural vocoder.

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:

### 2.3.3 Speaker embeddings

- **Sampling Rate:** Audio files were resampled to 22.05 kHz.
- **Silence Trimming:** Leading and trailing silence was trimmed to reduce computational waste and prevent the model from learning to generate excessive silence.

- **Mel-Spectrograms:** Mel-spectrograms were extracted using a frame size of 50 ms (1102 samples) and a hop size of 12.5 ms (276 samples). The number of Mel-filterbank channels was set to 80.

- **Pre-emphasis:** A pre-emphasis filter with $\alpha = 0.97$ was applied before the Short-Time Fourier Transform (STFT).

The acoustic features extracted for training were 80-band Mel-spectrograms computed using a Short-Time Fourier Transform (STFT) with a window size of 1024, a hop length of 256, and frequency limits of 0 Hz to 8000 Hz.

## 2.4 Model Architectures and Configuration

Two distinct acoustic models were trained on each of the three datasets, resulting in a total of 6 experimental models.

All models were implemented and trained using the **Coqui TTS** [28] open-source framework, which provides robust implementations of state-of-the-art TTS architectures.

### 2.4.1 Tacotron 2 Configuration

The autoregressive model used is **Tacotron 2** with the Dynamic Convolutional Attention (DCA) mechanism to speed up alignment convergence.

- **Encoder:** A 3-layer convolutional stack followed by a bidirectional LSTM (512 units).

- **Decoder:** A 2-layer LSTM (1024 units) with location-sensitive attention.

- **Speaker Conditioning:** d-vectors were not used for Tacotron 2. Instead, a learnable speaker embedding layer (dimension 256) was concatenated with the encoder output. This allows the model to optimize the speaker space specifically for the training set.

### 2.4.2 FastPitch Configuration

The non-autoregressive model used is **FastPitch**.

- **Architecture:** Feed-forward Transformer with 6 encoder layers and 6 decoder layers.

- **Aligner:** Since FastPitch requires duration targets, the model was trained using an unsupervised alignment search (Soft-DTW) available in Coqui TTS, eliminating the need for an external aligner like the Montreal Forced Aligner.

- **Predictors:** Explicit pitch and duration predictors composed of 1D-convolutional layers were trained jointly with the model.

### 2.4.3 Multi-Speaker Conditioning

To enable multi-speaker synthesis, both Tacotron 2 and FastPitch were conditioned on speaker identity. The speaker identity was provided using fixed-length embeddings, specifically **x-vectors** [35]. These x-vectors were extracted using a speaker encoder model pre-trained on a large, external multi-language dataset (e.g., VoxCeleb) and were kept frozen during the TTS model training. The x-vector for each utterance was concatenated to the output of the respective acoustic model's encoder, allowing the decoder to condition the generated Mel-spectrogram on the target speaker's voice characteristics.

### 2.4.4 Vocoder

The acoustic models predict Mel-spectrograms, which require inversion to raw waveforms. For this task, a high-fidelity neural vocoder was employed. Consistent with the literature review's justification for language-agnostic generalization, the pre-trained **HiFi-GAN v2** [27] model was used for all experiments. The vocoder weights remained fixed throughout the TTS training, ensuring that differences in final audio quality stem only from the acoustic models' performance under the different data sampling strategies.

## 2.5 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.

*3 Tacotron 2 DCA 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 |

*4 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 |

## 2.6 Model training configurations

### 2.6.1 Tacotron 2 with DCA hyperparameters

### 2.6.2 FastPitch hyperparameters

## 2.7 Training Procedure

### 2.7.1 Computational resources

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

### 2.7.2 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.

### 2.7.3 Hyperparameters

All models were trained for a maximum of 200 epochs ( 90,000 steps) or until convergence plateaued.

- **Batch Size:** Set to 64 for Tacotron 2 and 32 for FastPitch (due to higher memory constraints of the Transformer attention maps).
- **Optimizer:** The AdamW optimizer was used with $\beta_1 = 0.9$ and $\beta_2 = 0.998$.
- **Loss Functions:**
  - **Tacotron 2:** Post-net MSE, Decoder MSE, Guided Attention loss, and Stopnet binary cross-entropy loss.
  - **FastPitch:** Mel-spectrogram MSE, duration predictor loss (MSE), and pitch predictor loss (MSE).
- **Stopping Criteria:** Models were trained for a maximum of 200,000 steps. Early stopping was applied if the validation loss did not improve for 10,000 consecutive steps. Training time stability and attention alignment convergence were monitored, particularly for the low-depth Strategy C, as these metrics provide insights into the models' robustness under data scarcity.

## 2.8 Evaluation protocol

The synthesized speech from the trained models was evaluated using a combination of objective and subjective metrics.

### 2.8.1 Test set

### 2.8.2 Objective Metrics

Objective metrics such as the Mel-Cepstral Distortion (MCD) and attention alignment failure rate (for Tacotron 2) were calculated on a held-out test set (5% of the total data, unseen during training). While useful for monitoring training, these metrics do not perfectly correlate with human perception and served primarily as diagnostic tools.

### 2.8.3 Objective Analysis

In addition to MOS, the training stability was monitored. specifically for Tacotron 2. The **attention alignment plots** were generated at regular checkpoints. A failure to converge to a diagonal alignment indicates that the model has failed to learn the text-to-audio mapping. This binary metric (Converged / Failed) is crucial for the *Set-Breadth* (7.5 min/speaker) scenario, where data sparsity may prevent attention mechanisms from stabilizing.

### 2.8.4 Subjective Mean Opinion Score (MOS)

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

A group of 20 native Lithuanian speakers was recruited to participate in the evaluation. The listening test followed a Latin Square design to mitigate ordering effects.

- **Rating Scale:** Naturalness was rated using the standard 5-point Likert scale (1=Bad to 5=Excellent).
- **Test Design:** A **Latin square design** was employed to mitigate order and repetition biases. The experiment included a set of 20 unique test sentences, synthesized by all experimental models (4 models: Tacotron 2 and FastPitch, each trained on three strategies, plus a baseline human recording).
- **Raters:** 20 native Lithuanian-speaking raters were recruited for the listening test. Each rater evaluated a randomized block of sentences, ensuring balanced exposure to all models and sentences. The final MOS was calculated as the arithmetic mean of all collected ratings for each model.

The web-based interface used for the evaluation presented samples in a randomized order, ensuring no rater could identify which model or dataset produced the audio.

# 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] Vilnius University. *Lietuvių šneka valdomų paslaugų plėtra - LIEPA 2 (Development of Services Controlled by Lithuanian Speech - LIEPA 2)*. Project funded by the European Regional Development Fund. Available at `https://xn--ratija-ckb.lt/liepa-2/apie-projekta-liepa-2/`. 2020.

[2] C. E. Shannon. "Communication in the Presence of Noise." In: *Proceedings of the IRE* 37.1 (1949), pages 10–21.

[3] S. Jothilakshmi, V. Gudivada. "Chapter 10 - Large Scale Data Enabled Evolution of Spoken Language Research and Applications." In: *Cognitive Computing: Theory and Applications*. Edited by V. N. Gudivada, V. V. Raghavan, V. Govindaraju, C. Rao. Volume 35. Handbook of Statistics. Elsevier, 2016, pages 301–340. `https://doi.org/https://doi.org/10.1016/bs.host.2016.07.005`. URL: `https://www.sciencedirect.com/science/article/pii/S0169716116300463`.

[4] D. Gabor. "Theory of communication. Part 1: The analysis of information." In: *Journal of the Institution of Electrical Engineers - Part III: Radio and Communication Engineering* 93 (26 1946), pages 429–441. https://doi.org/10.1049/ji-3-2.1946.0074. URL: https://digital-library.theiet.org/doi/abs/10.1049/ji-3-2.1946.0074.

[5] 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.

[6] 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.

[7] 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.

[8] A. Black, N. Campbell. "Optimising Selection Of Units From Speech Databases For Concatenative Synthesis." In: 1 (1996).

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

[10] 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. https://doi.org/10.1109/JPROC.2013.2251852.

[11] R. Sproat, A. Black, S. Chen, S. Kumar, M. Ostendorf, C. Richards. "Normalization of Non-Standard Words." In: *Computer Speech and Language* 15 (2001), pages 287–333. https://doi.org/10.1006/csla.2001.0169.

[12] M. Bisani, H. Ney. "Joint-sequence models for grapheme-to-phoneme conversion." In: *Speech Communication* 50.5 (2008), pages 434–451. ISSN: 0167-6393. https://doi.org/https://doi.org/10.1016/j.specom.2008.01.002. URL: https://www.sciencedirect.com/science/article/pii/S0167639308000046.

[13] P. Kasparaitis, D. Antanavičius. "Investigation of Input Alphabets of End-to-End Lithuanian Text-to-Speech Synthesizer." In: *Baltic Journal of Modern Computing* 11 (2023). https://doi.org/10.22364/bjmc.2023.11.2.05.

[14] V. M. University. *Kirčiuoklis*. URL: https://kalbu.vdu.lt/mokymosi-priemones/kirciuoklis/ (viewed 2025-12-07).

[15] T. Mikolov, K. Chen, G. Corrado, J. Dean. *Efficient Estimation of Word Representations in Vector Space*. 2013. URL: https://arxiv.org/abs/1301.3781.

[16] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, et al. *Scaling Laws for Neural Language Models*. 2020. URL: https://arxiv.org/abs/2001.08361.

[17] I. Sutskever, O. Vinyals, Q. V. Le. *Sequence to Sequence Learning with Neural Networks*. 2014. URL: https://arxiv.org/abs/1409.3215.

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

[19] 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.

[20] J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, Y. Bengio. *Attention-Based Models for Speech Recognition*. 2015. URL: https://arxiv.org/abs/1506.07503.

[21] 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.

[22] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin. *Attention Is All You Need*. 2023. URL: https://arxiv.org/abs/1706.03762.

[23] J. Kim, S. Kim, J. Kong, S. Yoon. *Glow-TTS: A Generative Flow for Text-to-Speech via Monotonic Alignment Search*. 2020. URL: https://arxiv.org/abs/2005.11129.

[24] 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).

[25] 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. https://doi.org/10.1109/TASSP.1984.1164317.

[26] 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).

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

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

[29] 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.

[30] J. Lorenzo-Trueba, T. Drugman, J. Latorre, T. Merritt, B. Putrycz, R. Barra-Chicote, A. Moinet, V. Aggarwal. *Towards achieving robust universal neural vocoding*. 2019. URL: https://arxiv.org/abs/1811.06292.

[31] S. Arik, G. Diamos, A. Gibiansky, J. Miller, K. Peng, W. Ping, J. Raiman, Y. Zhou. *Deep Voice 2: Multi-Speaker Neural Text-to-Speech*. 2017. URL: https://arxiv.org/abs/1705.08947.

[32] Y. Jia, Y. Zhang, R. J. Weiss, Q. Wang, et al. *Transfer Learning from Speaker Verification to Multispeaker Text-To-Speech Synthesis*. 2019. URL: https://arxiv.org/abs/1806.04558.

[33] A. Nagrani, J. S. Chung, A. Zisserman. "VoxCeleb: A Large-Scale Speaker Identification Dataset." In: *Interspeech 2017*. ISCA, 2017. https://doi.org/10.21437/interspeech.2017-950. URL: http://dx.doi.org/10.21437/Interspeech.2017-950.

[34] 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)*. 2014, pages 4052–4056. https://doi.org/10.1109/ICASSP.2014.6854363.

[35] 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)*. 2018, pages 5329–5333. https://doi.org/10.1109/ICASSP.2018.8461375.

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

[37] 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.