DEPARTMENT OF PHILOSOPHY, LINGUISTICS AND THEORY OF SCIENCE

SPEECH SYNTHESIS AND RECOGNITION FOR A LOW-RESOURCE LANGUAGE

Connecting TTS and ASR for mutual benefit

Liliia Makashova

Master's Thesis:	30 credits
Programme:	Master's Programme in Language Technology
Level:	Advanced level
Semester and year:	Spring, 2021
Supervisor:	Asad Basheer Sayeed, Sjur Nørstebø Moshagen, Brendan Molloy
Examiner:	Staffan Larsson
Report number:	(number will be provided by the administrators)
Keywords:	Speech synthesis, automatic speech recognition, low-resource language, machine learning, transfer learning

Abstract

Speech synthesis (text-to-speech, TTS) and speech recognition (automatic speech recognition, ASR) are the NLP technologies that are the least available for low-resource and indigenous languages. Lack of computational and data resources is the major obstacle when it comes to the development of linguistic tools for these languages.

We present a framework that does not require enormous GPU and target data resources, as well as guarantees reasonably good results in performance for the end-product. In this work we perform dual connection between TTS and ASR models and make them learn from each other in a low-resource setup. This project, being the first open-source implementation of such a bidirectional algorithm, leverages the power of open-source projects for the benefit of indigenous languages. We release the first ever functioning ASR tool for the North Sámi language along with a competitive TTS technology, which fulfills the demand of the North Sámi community and globally contributes to the further development of AI tools for low-resource languages.

Acknowledgments

First and foremost, I want to thank Brendan Molloy from The Techno Creatives for always being incredibly helpful, any time and any day. I also want to thank Sjur Nørstebø Moshagen, Børre Gaup and Katri Hiovain-Asikainen for their invaluable feedback and encouragement, as well to the whole Divvun team. I want to express my gratitude to my academic supervisor Asad Basheer Sayeed for his important advice and guidance. Also, this project would not be possible without my family and friends supporting and believing in me.

Resources for the completion of this project and a student stipend were provided by The Techno Creatives and Divvun.

Contents

1. Introduction

1.1 Focus and area of the project

With more than 7000 languages in the world, only a tiny number of them benefit from the development of language technologies. Only less than a dozen of the world's languages have gone through consistent and expensive methods of creating and developing technology around them. These languages—like English, Chinese, Spanish, French, German—are, of course, spoken by a large number of people in the world. For them such advances like next-word suggestion, spelling and grammar correction, machine translation or talking to a virtual assistant, have already become a part of everyday life and are not a novelty anymore.

However, this is not the case for the languages that have not yet received enough attention from the private sector. For those, it is now a task left for national governments to invest large funds in the technologies where their languages will be used (Cooper, 2017).

Depending on the branch of language technology, the task of filling in the gap might look more feasible than in some other cases. For example, for some languages it is enough to just have a large set of printed text corpora (e.g. novels, newspapers) to build a working next-word prediction model. This project, however, deals with one of the most difficult tasks that concerns the development of technology for low-resourced languages—speech synthesis and speech recognition.

This task is one of the most difficult ones, because of the expensive kind of data needed for this task (Cooper, 2017, de Korte, 2020). Text-to-speech (TTS) and (automatic) speech recognition (ASR) tasks require: 1) a lot of data and 2) good quality data (audio samples recorded in professional sound studios by professional voice actors). Having these two available, Google's Assistant, Apple's Siri and Amazon's Alexa were created. One of the most important uses of the TTS and ASR tasks is not only communicative, but also assistive (e.g. accessibility technology for visually impaired people). There are some impressive examples of progress made in this direction from big corporations like Apple and Google, but their progress concerns only several dozen of the world's languages and even accent sensitive systems were made for large western-European ones—several voices (dialects) for English, Spanish, French and Portuguese with a total of 30 languages for Apple devices, and much more (with different functionality coverage)—a total of 285 languages for Android ("iOS and iPadOS - Feature Availability", n.d., 'Language support. Cloud Speech-to-Text Documentation", n.d.).

Apart from requiring large amounts of good quality data, the task of executing algorithms for speech synthesis and/or recognition is also extremely expensive in terms of computational resources. GPUs being more expensive than usual during the timeline of this project and suffering from market scarcity (Molloy, D. 2021) poses one more challenge for the tasks of not only ASR and TTS technology but also for any machine learning project.

In addition, even having computational power available, it is also a question of responsibility and sustainability when it comes to the use of resources for machine learning tasks in times when the most known state-of-the-art models are also the most resource hungry ones (Strubell et al., 2019). In recent times there has also been a number of concerns raised about the balance between possible benefits from AI technology and AI's contribution to climate change and the enforcement of the "greenhouse effect". (Dickson, 2020)

1.2 Ethical considerations

There are many ethical concerns around AI technology nowadays (Rehm, 2020). Cultural appropriation is one of the most important ones, especially when working with indigenous and low-resource languages.

This project, however, is a part of Divvun group, whose team is responsible for the development of tools for the Sami community, and whose main task is to provide language assistive tools for indigenous languages. Therefore this project is being made *together* with the Sámi community and its aim is to fulfill the community's requests, not to exploit their cultural heritage.

Our project is also oriented on filling in the gap in *locally* produced language technology for small European and indigenous languages (Rehm, 2020). With the market being overwhelmed with English-oriented and not-locally produced technologies (e.g. United States - Siri, Alexa, Google Assistant etc.), the project complies with the EU's narrative, that developing language (AI) technologies for *all* European languages is one of the main demands on the European market.

1.3 Contribution of the project

This project implements end-user suitable speech recognition and speech synthesis systems for the North Sámi language, by leveraging a great number of open-source resources for both better performance and lower GPU consumption. This experiment is therefore much easier to reproduce in low-data and low-computational resourced environments and very fitting for a university or private individual setup. By using official open-sourced pretrained models from large companies like Nvidia and Facebook, we as well have an opportunity to conduct an experiment on fine-tuning and confirm the benefits of transfer learning for low-resource languages, at least in the case of developing ASR and TTS systems.

Instead of reimplementing models that are already available for anyone willing to use them, this project allows us to focus on the implementation of the unique part of this work: the dual connection between speech recognition and speech synthesis models. The process that we conduct—Dual Transformation—is described only in a couple of academic papers and was applied within a somewhat lower-resource setting. Unlike previous projects concerning this dual connection, this work implements this algorithm from scratch in an ultra-low-resource setup. With this we show a dual connection between ASR and TTS systems and, at the same time, combating multiple issues relevant for the low-resource setup.

1.4 Outline

The previous research and the current state of development of ASR and TTS technologies are briefly described in Section 2. The next section, Section 3, describes in more detail the setup and the structure of this project. Section 4 (Resources) contains the description of the datasets used in this project. Section 5 describes the principles of the machine learning algorithms (models) used to make this project possible. Section 6 (Experiments, Implementation and Reproducibility), should be considered and can be used as a more practical read for one aiming to reproduce this project. Section 7 (Results) and Section 8 (Conclusions and future work) discuss the results and possible extension of the project.

This project is submitted with the accompanying and maintained GitHub repository, which one should refer to for more technical reference[1].

[1] Project implementation is available at Divvun's official Github repository: https://github.com/divvun/lang-sme-ml-speech.

2. Background and Previous work

Attempts to implement TTS and ASR technologies have been around even since the beginning of the 20th century. In the 1930s, Bell Labs developed the vocoder—a device which automatically analyzed speech into its fundamental tones and resonances. Then, a keyboard-operated voice-synthesizer called The Voder (Voice Demonstrator) was built by Homer Dudley (Gold, 2011). The first English TTS system was developed at the end of the 1960s by Norico Umeda in Japan (Klatt, 1987). Then, in the 1980s hidden Markov models (HMM) became popular to use in this space.

During the development of TTS technology two main branches were formed—concatenative and formant synthesis. The first one at its core uses concatenated parts of pre-recorded speech to produce a new output, and the second one does not use any pre-recorded human speech, but aims to generate an artificial voice output by adjusting the waveform produced by an acoustic model. (Sciforce, 2020)

Then, neural-based TTS synthesis came into the game and since mid-2000 TTS systems acquired a very complex structure that would usually consist of a text frontend (with linguistic features), an acoustic model and some kind of signal processing vocoder. These system components would be trained separately and therefore had more chances to produce errors.

Various research has shown that when it comes to TTS and ASR, the results of the project implementations must be close to perfect. This is because for the target user of this technology, the two main requirements for the final product are naturalness and intelligibility. Several experiments with not-so-well performing TTS applications have demonstrated that users tend to be very excited about the introduced technology at the beginning of the trial, however they seem to have no desire to continue using it soon enough and indicate their annoyance about the imperfections of the product (Taylor, 2009).

Nowadays, a big focus within TTS technology is on the neural-based and end-to-end approach. Large contributions to this area are currently being made by big companies like Google, Facebook and Microsoft, who use this technology both in their own products as well as contribute to the academic research within this field.

In recent years the most popular approach to the TTS and ASR task is building end-to-end systems, so that a single model can take care of all the inner tasks. In 2016, DeepMind introduced a generative model for a TTS task named WaveNet. This state-of-the-art deep neural network showed good results when applied to TTS, ASR and music generations tasks. The WaveNet model focused on generating speech from mel-spectrograms—a representation of a short-term sound spectrum (Van den Oord et al., 2016). In 2017, work by Google called Tacotron was introduced. This deep learning model was able to generate audio from raw text (Wang et al., 2017). Similar results were attained also by Facebook with their project VoiceLoop (Taigman et al,. 2018). That same year, Google released the updated Tacotron2, which employed both WaveNet and modified Tacotron technology to design a state-of-the-art speech synthesis system (Shen et al., 2018).

Such end-to-end systems have their own pros and cons. The most obvious and biggest positive feature of these technologies is that they can be implemented using only one single model for generation of mel-spectrograms. This fact therefore contributes to better reproducibility and less erroneous performance. Also having only one model doing the entire job contributes to easier fine-tuning of the trained model. The switch between languages, speakers and even datasets can now be made very smooth. Finally, the results of these models are much better when it comes to naturalness and intelligibility of the produced audio output. The main disadvantage of these models, especially in the scope of this project, is that they still require a lot of training data. The recent models, in comparison to older ones, need less training data, however, the amounts they require usually still exceed the resources available for indigenous languages.

Developing a TTS and ASR tool which would be capable of producing natural and intelligible speech based on small data is the main research question of this project. The previous end-to-end models showed an ability to be trained on insufficient data, but their audio output suffered significantly in robustness (Ren, Tan et al., 2019, Xu, Tan et al., 2020). The academic research into NLP tools for low-resourced languages looks rather sparse as most of the research is still focused on languages like English and on fine-tuning of the existing models. Only a small number of recent works made a contribution to the scope of this project.

A more low-resource focused work has recently come from well-established Microsoft. This fact might show a trend that big companies are now starting to pay attention to the smaller languages. The work by Jin Xu, Tan et al. (2020) demonstrated a seemingly successful approach of using current end-to-end state-of-the-art models to implement TTS and ASR technologies for low-resourced languages. The achievement of reportedly good results was made possible due to combining training data from low-resource languages with a resource-rich one (usually English). In this way, a lot of features (like phonemes) are used from a pretrained model on the resource-rich language and then basically adjusted to fit the low-resource language on the second stage of the training.

As for ASR-only tasks, the absolute majority of state-of-the-art models were trained on thousands of hours of high quality data. One of the leaders in performance is Baidu's DeepSpeech model, which was trained on 12 000 hours of data (speech-transcription pairs) (Hannun et al., 2014). Facebook's Wav2Vec on the other hand, was trained on 53 000 hours of unpaired data (only speech) and also shows good performance, and according to its authors, has the perspective to be used for fine-tuning on target low-resource languages (Baevski et al., 2020).

3. Low-resource setup and our workflow

This project, being extremely low-resourced not only in terms of data, but also in terms of computational power, required a bit of a different workflow setup than is usually performed in ML experiments.

To tackle low-resource data issues, most of the achievements demonstrated in this project are the result of transfer learning training (not training from scratch), as this was the only way to fill in the gaps in data by transferring the parts that were already learned in resource-rich languages.

Only open source models (trained on resource-rich languages) were chosen for this project to tackle the GPU, workload and time-constraints issues. Even having publicly available large datasets (e.g. LJSpeech, LibriSpeech), the pre-training of models like Tacotron or Transformer on resource-rich languages would still be difficult. It would take a lot of resources from the main project—developing for the North Sámi—considering the complexity of those models and the computational requirements that they have. For example, most of the publicly available models were trained with a multi-GPU setup, which is not an option to use within this project.

Knowing the state of current research and being aware of our capabilities, this project concerns several aspects. As in Xu, Tan et al. (2020) work, we are connecting TTS and ASR models for North Sámi with common training procedures to be able to benefit from the dual nature of these two tasks. Xu, Tan et al.'s (2020) approach in its core proposes a flow where two models are firstly trained on resource-rich languages and then are fine-tuned on the target (low-resource) language. And within the fine-tuning stage, the two models are trained together.

Our main **research question** is whether Xu, Tan et al. (2020) approach can in fact be a solution for *any* low-resource language and whether this approach can work with *any* state-of-the-art pretrained model. To specify, this project investigates whether 1) it is possible to implement TTS and ASR systems with data thousands of times smaller than in the state-of-the-art models, 2) it is possible to use rich-resource languages (like English) for the benefit of low-recourse ones (like Sámi) and 3) it is possible to make two models learn and benefit from each other.

With our ASR/TTS connection loop we are adopting Xu, Tan et al. (2020) general architecture in part of the Dual Transformation loop, which (as far as we are informed) is now the first open-source implementation of this kind of dual task.

To briefly describe our workflow, to transfer-learn we use Tacotron on the TTS-side of our project and Facebook's Wav2Vec on the ASR-side of our project. We make these two sides of the system learn from each other during the training with a dual-transformation loop (shown in Figure 2 in Section 5).

The second concern of this project is to make an end-to-end model that would have a potential to be deployed for the end-user. Apart from having good performance, we also considered the legal issues. All base-models for this project were chosen with the consideration of having a free licence (e.g. MIT, BSD 3-Clause), being efficient and showing fast inference time. Apart from these aspects, we have modified the default inference scripts and adjusted saving mechanisms (where it was needed) to make the final weight of our product even lighter, however, still efficient.

4. Resources

4.1 Data for TTS & ASR

The cost of training data for speech synthesis and recognition is high. This is mostly due to the fact that for the industrial deployment of TTS and ASR services single- and multi-speaker high quality recordings in a professional studio are needed. TTS requires fewer in number but higher in quality data, while ASR can be performed with lower (in comparison, but definitely not low!) quality of audio data, but its amount should be several times bigger than for TTS. Having multi-speaker data for ASR training is a key requirement, as it is the best way currently possible to generalize to an unseen speaker (Ren, Tan et al., 2019, Xu, Tan et al., 2020).

The mechanism inside TTS and ASR algorithms can be described as having a reversed relation. In usual practice, a TTS algorithm, built on paired data (text and audio), uses text normalisation tools, the grapheme to morpheme conversion (often made with pronunciation lexicon) and a speech model that converts a model's output into an audio file. In reverse to this, an ASR algorithm firstly uses an audio perception module to perceive speech (input audio), then converts phonemes to morphemes and only then uses text tools to produce a written output (Figure 1).

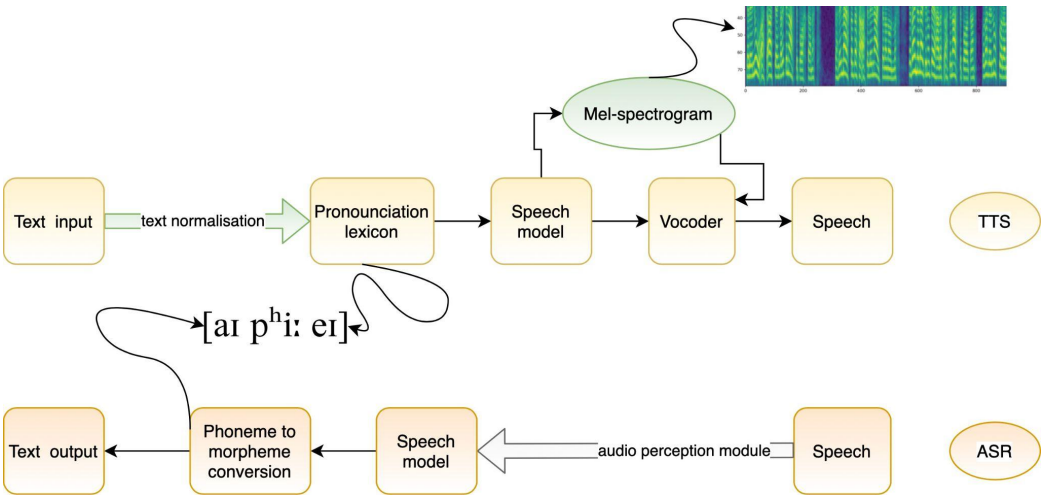


Figure 1. Schematic representation of the TTS and ASR models' general mechanisms.

Unpaired data, which is not the main training resource for these tasks can also sometimes be used for fine-tuning purposes. Such data, which is easier to acquire, is usually used to develop some working tools for low-resourced languages as resources for annotation (pairing) are limited. However, in the unpaired setting it is more difficult to reach a good quality of the speech/text produced (Cooper, 2017). Paired data in big amounts is used for training TTS and ASR for resource-rich languages and usually shows better performance on inference level. Morpheme to phoneme dictionary for TTS which is important for better pronunciation quality is usually not available for low-resourced languages.

Many researchers have been trying to prove that all three components (paired data, unpaired data and a pronunciation lexicon) are needed for a good quality final product. In the case of low-resourced languages, the quantity of data for all three components could be shrunk but should not be missing. Thus, paired data, unpaired data and a pronunciation lexicon should be acquired. (Ren, Ruan et al., 2019)

Having this statement, the cornerstone of this project was the absence of a pronunciation lexicon (e.g. ARPAbet for English) for the North Sámi language (at least within resources of this project). Therefore, the approach of the already mentioned Xu, Tan et al. (2020) was chosen to conquer the task.

4.2 Target dataset

As for this project, the data available consists of around 3 000 audio files recorded with one female and one male voice actor (Sme-speech, provided for this project by Divvun&Giellatekno). All the files make up around 3 hours of speech per speaker. Each audio file is accompanied by a text file containing the extracted transcription of the utterance. The project dataset possesses a few discrepancies that were considered while implementing the project. The accompanying text files have been modified according to the actual reading of the voice talents. Therefore the text files for male and female voices are not identical. The texts follow the rules for standard orthography in general. However, if the reading of the voice talent was inaccurate, the text has been modified accordingly. For example, if an 'nc' consonant cluster is pronounced more like 'nzz', the word in our data was altered according to the reading, which results in the fact that now the word is no longer recognized by any of the available spellers.

In addition, both voice talents speak the Guovdageaidnu dialect of North Sámi. The reading chosen for the TTS project is a conservative Guovdageaidnu dialect. Some distinctions that are lost in less conservative variants are therefore retained in the recordings, such as the difference between: 'žž' and 'ddj', 'čč' and 'dj', and 'sk' and 'tk'. The talents were instructed to retain these and other distinctions in their readings. It also requires mentioning that the development dataset for this project is not open source. The Sámi Parliament of Norway retains full rights to the sound files. The recordings cannot be used without the written consent of the Sámi Parliament and therefore this project was implemented within the agreement with the Sámi Parliament and its results cannot be reproduced without communicating with the rights holders. Although we can guarantee the reproducibility of the *results* of this project, we do not provide the data, and if one is willing to repeat our experiments they should use any other (their own) dataset of similar size and structure or seek permission for the original dataset from the Sámi Parliament.

The quality of data for this project can be considered somewhat satisfactory. The recordings were made at a professional studio in 2011. However, the noise level of the recordings is not consistent across speakers and subfolders of data. The perfect solution for this would be to exclude outstanding recordings from the training data, however, having less than 6 hours of data in total, we decided not to do this as all recordings, even the bad quality ones, are very precious for this project.

Apart from 6 hours of paired data, we have used a small subset of unpaired data. For speech synthesis, our texts without audio pairs come from the free corpora of North Sámi language—most of the texts are coming from municipalities and government documents (Sme-freecorpus, provided by Divvun&Giellatekno). As for ASR, we did not have truly unpaired and processed audio data (that could be retrieved from, for example, openly available podcasts or radio programmes, but were not due to time constraints). Instead, on the final stage of our project, we received additional audio data, retrieved from the long (around 5 minutes each) recordings, that we couldn't use for training. These recordings we use as unpaired data, ignoring their text pairs.

4.3 Additional Datasets

As was mentioned before, this project strongly relies on rich resource languages for pretraining TTS and ASR models. High quality English language datasets were used for initialisation of the models. Thus, we transfer language knowledge from English to the North Sámi and we refer to English language datasets as additional datasets.

For the text-to-speech task, the LJSpeech dataset was used. It is a public domain English speech dataset consisting of around 13,000 short (from 1 to 10 seconds) audio clips of a single female speaker. The total length of all audio material is approximately 24 hours (Ito, 2017).

For the speech recognition task, a large-scale high quality LibriSpeech dataset was used. This dataset is a multi-speaker (almost 2 500 speakers) corpus of read English speech with total lengths of audio files of

almost 1000 hours. LibriSpeech is balanced in terms of gender and per-speaker duration (Panayotov et al., 2015). A transcription accompanies each clip in both LJ and LibriSpeech datasets.

These two datasets were used for training of the open-source models that we use in this project as our base models.

	TTS (paired)	ASR (paired)	TTS (unpaired)	ASR (unpaired)
LJ Speech	✓			
LibriSpeech		✓		✓
Sme-speech (Divvun & Giellatekno)	✓	✓		✓
Sme-freecopus (Divvun & Giellatekno)			✓	

Table 1. Data for the project (including the datasets used for pretrained models).

5. Methods

Sequence-to-sequence learning is a core computational method of this work and transfer learning is a core implementation method of this work.

5.1 Sequence-to-sequence learning

Sequence-to-sequence (S2S) learning is applicable in many tasks of ML and AI, but in this section, the description of this method will be made with consideration of the tasks of this work.

In S2S setup, there are two domains, *X* and *Y*. Therefore, *x* and *y* are sequences that belong to X and Y domains respectively. We will call *X* a source domain and *Y*-target domain.

In TTS scenario, *x* would be a text sequence (array of phonemes or graphemes) of defined length *n*, e.g. $x = (x_1, x_2, \dots x_n) \ \& \ x \in X$. Corresponding to this, *y* would be an audio related sequence (mel-spectrogram as feature in this case) of defined length *l*, e.g. $y = (y_1, y_2, \dots y_l) \ \& \ y \in Y$. Then, a conditional probability is estimated via a parameter θ, which will be learned by the sequence-to-sequence model so that $P(y|x; \theta)$. The ASR scenario would look reversed to the TTS one, so that the source domain (*X*) would be an audio representation (mel-spectrograms) and the target (*Y*) domain would be text representation (phonemes or graphemes).

Sequence-to-sequence models are usually developed on the encoder-decoder structure, and this work is not an exception. This project uses an attention enriched encoder-decoder model. In this setup, the encoder takes as an input the features from the source domain and produces (encodes) its representation. After this, the decoder's job is to estimate the conditional probability (P) of the target element from the encoder's representations. In between these steps, the attention mechanism is used to allow the decoder to better learn important parts of the encoder's representations (Ren, Tan et al. 2019, Xu, Tan et al., 2020).

Further subsections more closely describe details of sequence-to-sequence modeling for each of the tasks of this project.

5.1.1 End-to-end TTS

The final implementation of the E2E model to generate audio from a text sample relies on four state-of-the-art open-source models. It employs the models of Tacotron, Tacotron2, ForwardTacotron and WaveGlow. This approach proved to be the most efficient and produced the best results (more on this in the Experiments, Implementation and Reproducibility section).

Tacotron and Tacotron2 are in essence end-to-end models for a TTS task. Tacotron and Tacotron2, unlike pure Transformer, are RNN based. They use a bi-directional LSTM encoder and a one-directional decoder, which are connected with location sensitive attention mechanisms (Wang et al. 2017, Shen et al. 2018). By design, employing a heavy computational (autoregressive) process, Tacotrons have a few disadvantages when it comes to deploying it for real-life applications. The most prominent of those would be slow mel-spectrogram generation which is needed for both inference and training. In addition, since each new mel-representation can be generated only with the regard to the previous one, some errors would inevitably be transferred throughout the training process, and the end result of the mel-spectrograms (and therefore the speech produced) sometimes would suffer from mispronunciations, word-skippings or repetitions. (Ren, Ruan et al. 2019)

Nevertheless, according to comparative studies, Tacotron2 has proved to be performing better than Transformer-TTS, when it comes to attention alignments (Karita et al., 2019). Using short-time Fourier transform (STFT) magnitude to predict mel-representation of text input, the attention alignments are more

stable in Tacotron2 (e.g. more diagonal)—which implies that it is easier to get closer to the real world character-to-phoneme alignment with this model. The experiment phase of this project has also confirmed that the alignments learned by the Tacotron and Tacotron2 models were reasonably good—the speech produced was intelligible enough.

Nowadays, the default vocoder technique to test any TTS algorithms is a Griffin-Lim algorithm. Being non-neural based it allows for quick inference at any stage of the TTS model training. It requires only a mel-spectrogram (or STFT magnitude representation) as its input, and by applying filters on it on each iteration, it returns a reconstructed audio file (Griffin et al. 1983). With GLA it is not possible to adjust the output with any specific features (like pitch, speed, etc.) but *is* possible to judge the model's progress in terms of grapheme-to-phoneme alignments to some extent. By 'some extent' we mean that during the training it is very possible to receive a not-consistent mel-representation and since GLA expects it to be consistent, the output usually will suffer from a lot of noise. In addition, the number of iterations required to produce a good output from GLA might be inconsistent throughout the model's outputs. Therefore GLA was used only for draft-testing while training and was not utilized for the final demonstration of the project's results.

The WaveGlow vocoder model is not a necessary part of this project and should be considered as an additional part on top of the TTS model. This project uses a fine-tuned WaveGlow model which was not modified structurally and was used only for final demonstration reasons to handle noisy output produced by GLA.

WaveGlow is used here to demonstrate to what extent it is possible to develop the project and allows for further work on the project in the direction of its deployment for the end-user, because only with a neural vocoder it is possible to reach human intelligibility and naturalness of speech in a low-resource setup. As it was mentioned before, a vocoder is a network usually used in the second neural step of the TTS modeling architecture (see Figure 1). More specifically, it is a network which uses mel-representations as its input (which are generated by sequence-to-sequence models as Tacotron or Transformer). The vocoder networks can be autoregressive or non-autoregressive, and can employ different computational methods (for example, being CNN based or GAN based). WaveGlow is a non-autoregressive model that can generate audio output within a single forward pass. The computation inside this model is a transformation from simple distribution conditioned by mel-spectrogram representation. This transformation uses coupling layers, convolutions, and early outputting (for propagation on early states of training) as its methods (Prenger et al., 2019). For a full explanation of the WaveNet algorithm, please refer to the project's paper.

ForwardTacotron, which is used as the final wrapping part of E2E TTS model, was chosen for this project due to 1) its training efficiency considering both a low-resource setup in terms of data and computational power, 2) it being compatible with Tacotrons which on the fine-tuning stage (see Experiments section) showed promising results 3) it being available publicly with a free licence. ForwardTacotron being non-autoregressive allows for smaller training time and faster inference. It is structurally similar to the FastSpeech project (Ren, Ruan et al., 2019), which implements a feed-forward Transformer network (meaning, without autoregression). ForwardTacotron is a modified implementation of Tacotron. The mechanism inside it allows for faster training by not using any attention for the generation of mel-spectrograms and substitutes the attention pass in Tacotrons with a feed-forward network. In the end, ForwardTacotron is an encoder-decoder model without an attention mechanism (Schäfer, n.d.).

5.1.2 End-to-end ASR

For speech recognition tasks, the Wav2Vec2 model from Facebook was used. At its core, it is a Transformer model, whose training was split into self-supervised and supervised stages. In this setup, the self-supervised part of training (referred to as pre-training by authors) takes 53 times more data (unpaired) than the supervised training (with paired data). Wav2Vec2 authors, to motivate such an algorithm for an ASR task, said that 'learning purely from labeled examples does not resemble language

acquisition in humans: infants learn language by listening to adults around them—a process that requires learning good representations of speech' and added that 'self-supervised learning has emerged as a paradigm to learn general data representations from unlabeled examples and to fine-tune the model on labeled data'.

Inside the Wav2Vec, the Transformer module precedes a multi-layer convolutional feature encoder and quantization module, which outputs latent speech representation. The Transformer model uses 'less attention' (Baevski et al. , 2020) by using lightweight convolutional layers, which authors call relative positional embedding.

By its design, Wav2Vec is one of the most suitable models to use for transfer learning purposes. This is due to the so-called cross-lingual approach and a minimum amount of paired data. During the longest phase of the training (with unpaired data) the model is trained to understand the smallest speech units which are shorter than a phoneme (25ms each). Therefore, each phoneme learned would eventually consist of several such speech units. Since many languages can share the same phonemes, they therefore would share the same speech units, and a model trained for one language can be beneficial to another one.

Since this model was not implemented from scratch and was only fine-tuned to make it ready for training with the dual task, please refer to the full explanation of the model's architecture from the Wav2Vec2 paper.

5.1.3 Dual task

Having a fine-tuned Speech Recognition and TTS models, the next step of this project is to benefit from the dual nature of these two tasks. Dual connection, according to (Tjandra et al., 2017, Ren, Tan et al., 2019, Xu, Tan et al., 2020) can significantly improve the performance of both TTS and ASR models.

The idea of dual transformation was first introduced by Ren, Tan et al. (2019), and then referred to in the work by Xu, Tan et al. (2020) as a suitable algorithm for low-resource projects (both works are coming from Microsoft). At its core, dual transformation combines supervised and unsupervised training on paired and unpaired data. Tjandra et al. (2017) have earlier proposed a somewhat similar idea, but the chained training of TTS and ASR model was made after both of the models were already fully trained.

The core idea of dual transformation is closely related to the recent ideas in the Artificial intelligence field, which tries to develop algorithms mimicking how human cognition works. The core idea of any ML algorithm is similar to how children learn new things—targeted repetition (Schwab et al., 2017)—whether it is just a repetitive exercise to be better at calligraphy (when learning to write) or the same kind of data split in equal batches repeated in every epoch (in machine learning).

However, there are many theories in psychology, and many of them assume that the ability to speak and understand the language is inborn to all humans, while the ability to read and write is acquired and can be considered as a part of any other learning process such as riding a bike or playing a musical instrument (Chomski, 1988).

The first language acquisition is a very complex cognitive process and our dual transformation algorithms cannot be compared to it. However, some methods of teaching a foreign language to kids (or even adults) have a lot in common with DT. To draw a line of comparison, in terms of human cognition we have an individual who is able to speak but has no understanding of a new language. In terms of ML we have a system that is designed to speak or write, but has not yet learned the alphabet, spelling and pronunciation rules of the target language.

Some of the methods of teaching a foreign language do not at all include learning of pairs like /new word - its translation/ ("What is the Berlitz Method", n.d.). Within this teaching, a student is expected to mostly repeat after a teacher, read foreign texts and receive feedback on their pronunciation, while not

understanding the meaning of what they have just spoken about. In ML terms, this method would be similar to training on unpaired data.

How the knowledge of a new language is eventually acquired by humans is out of the scope of this project. Instead, we will focus on the stage when words are learned and could be correctly (or almost correctly) spelled by the student, while there is still no background knowledge of what they mean.

For a technical explanation of how the DT loop is designed, please, refer to the next chapter (Experiments, Implementation and Reproducibility). The general algorithm is as follows: we prepare unpaired text data, which we use as input to our TTS model. The generated audio files are now becoming an input to our ASR model which continues to be trained with the loss value being calculated with regard to the original text input. Reversely, we need unpaired speech data to be an input to our ASR model. The output produced by the ASR model becomes an input to the TTS model (see Figure 3 for better understanding) and the loss value as well is being calculated with regard to the actual input speech.

This setup eventually gives a unique possibility to increase our corpus to continue training both speech-recognition and text-to-speech models while updating models' parameters on the fly.

5.2 Transfer learning

Transfer learning and fine-tuning are methods used in machine learning to get better performance of the final model. However, they have different names. In the literature they are often used interchangeably, which might lead to confusion when reading this paper as well. To clarify this, the main method used in this paper should be considered to be transfer learning, and fine-tuning is being a kind (a scenario) of transfer learning (PyTorch, n.d.).

Only one submodel in this project was trained from scratch, all the others (described in the previous subsections) models were acquired, had some of their layers frozen or reinitialized and then trained on the new data. These three steps define transfer learning.

Fine-tuning, on the other hand, is often explained to be only the final stage of the training process, as for example, adjusting the learning rate, adding more epochs or changing some other hyperparameters to make the model converge faster. This definition implies that fine-tuning is also a part of any training process, including training within a transfer-learned model. Therefore in this paper, we use both terms and by saying 'fine-tuned model' we mean that the model was fine-tuned after being transfer-learned.

For a visualisation about which ways the models are used and how the processes are connected, please refer to Figure 2.

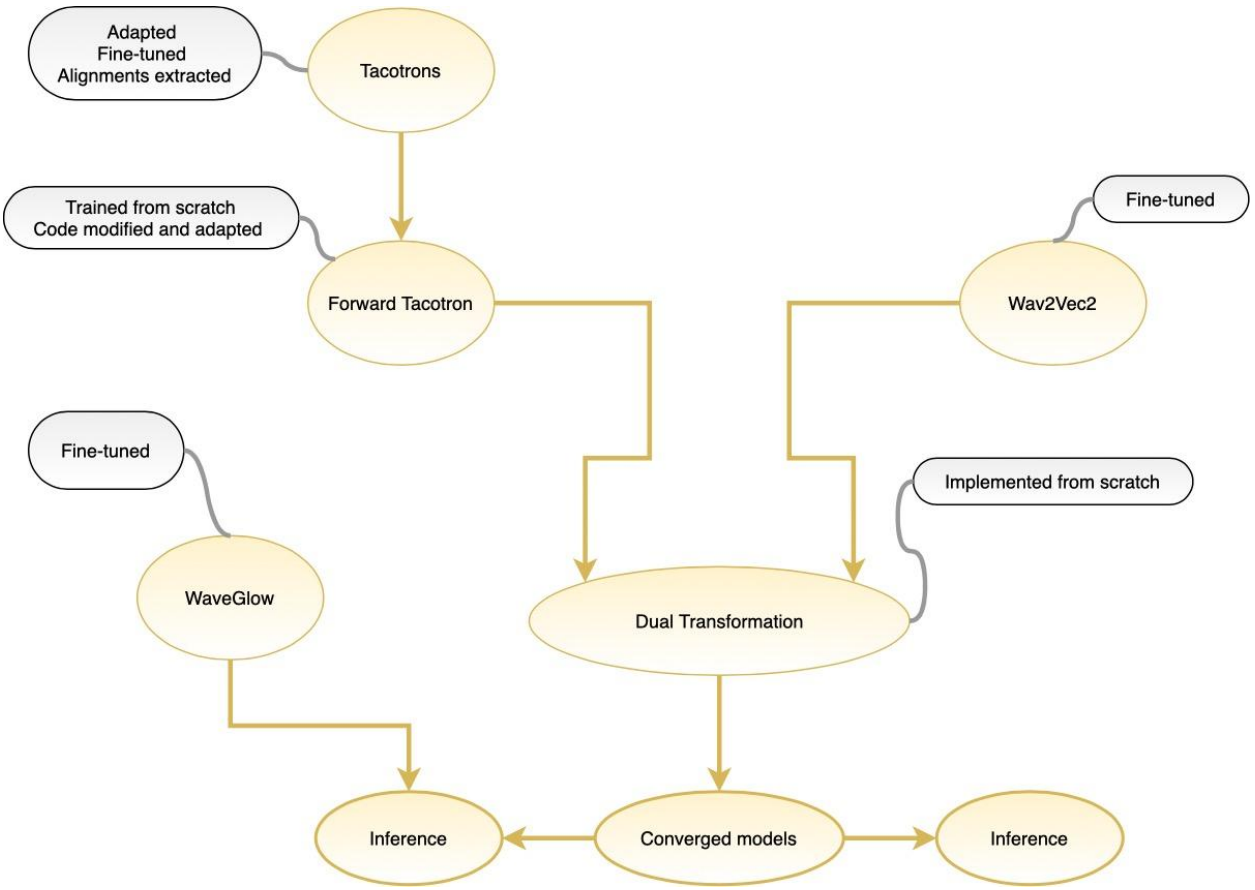


Figure 2. Models used for this project. Left side of the figure–text-to-speech algorithm, right–speech recognition algorithm.

6. Experiments, Implementation and Reproducibility

This chapter gives a detailed description of the implementation for the TTS/ASR model and the changes made to open-source parts that were used for this project. This chapter also presents instructions for reproducibility for those aiming to reproduce the results of this work.

6.1 TTS model

First, we preprocess the North Sámi data, by organizing it to have the same structure as the LJSpeech corpus has (Ito, 2017)—e.g. converting .flac files to .wav, .docx files to .txt and downsampling all audio files to 22 kHz (original data is 44,1 kHz). As the open-sourced Tacotron2 model was trained on a single speaker female voice, we as well are using only female voice recordings for fine-tuning of each of the models for TTS. We make the train/validation split with a ratio 3591/200. This project provides a Readme file in the accompanying repository. Please, refer to it for more details on preprocessing and training.

The text-to-speech model is the most complex part of this project, consisting of four parts—Tacotron and Tacotron2, ForwardTacotron and WaveGlow. Tacotron, Tacotron2 and ForwardTacotron are used for training the model to produce highly consistent speech representation for the input text—mel-spectrograms. Then, WaveGlow is used as a neural vocoder at inference step to transform raw mel-spectrograms into human-sounding speech.

The Tacotron2 and WaveGlow base models were used from the official Nvidia repository. The ForwardTacotron project was retrieved from Ideas Engineering's (a company that specializes in developing software for the media industry) official repository.

It needs to be mentioned that Tacotron2 *can* be used directly for inference with WaveGlow for speech synthesis (without the pass to ForwardTacotron). However, the experiments proved to be not very successful, as Tacotron2, because of its core design, takes a long time to converge and propagates pronunciation errors that are difficult to fix on inference level or control during training.

During the experiments and after some time spent on training, while experiencing issues with performance, we started to look for solutions for errors we encountered. A profound difference of Tacotron2, compared to the first version of it (referred further as Tacotron1), was avoiding *reduction window(factor)* setting, while still using teacher forcing. Reduction factor is an important feature that helps the model pick the alignments up faster, reduces training and inference time. To clarify, teacher forcing is the method used in training, when the output of the model from the previous step is considered by the model as a ground truth value (along with actual training data), therefore, the output becomes an input. This method is considered beneficial for Sequence-2-Sequence language models, as it enhances their sequential nature, however, it may lead to slow convergence and to the model behaving unstably, which is what we saw in our case. Reduction window, in its turn, makes the teacher-forced input consistent with the target one, as we would repeat such an input the number of times that corresponds to the reduction factor value. In other words, the RF value indicates how many teacher-forced inputs we need to ignore during the training at equal intervals of steps. This, therefore, increases the distance between the model's and the actual outputs a.k.a. Euclidean distance, and makes the output of the model depend more on the actual text input, than on the output from the previous time step.(Liu et al., 2019)

Since this project is constrained by defined time-frames, and we needed to return now to Tacotron1 to confirm our assumptions about the reasons for persistent errors, we re-fine-tune our fine-tuned Tacotron2 model with Tacotron1 training mechanism (meaning, with reduction window). With both Tacotrons having such a similar structure, we manage to migrate the code and adjust the training only by making a few modifications.

After the Tacotron1 model is trained (fine-tuned on Sámi), it is used to extract attention alignments for the target data (the North Sámi speech corpus). Then ForwardTacotron takes these alignments as training material. It, however, does not use any attention by itself when training.

As for adaptive changes made to the open-source implementations used for this project, the ForwardTacotron model, as well as both Tacotron models were modified to use character (grapheme) embedding, instead of phonemes, as there was no available pronunciation lexicon for the North Sámi. When fine-tuning, the embedding layer of the pre-trained Tacotron2 is ignored and the transfer-learned model is now having a different size of this layer (corresponding to the grapheme array length). ForwardTacotron in its original implementation also uses pitch conditioning for generating more human-like audio, which had to be removed in this project as it is currently not possible to implement this part for Sámi languages.

With these modifications, Tacotrons were fine-tuned and ForwardTacotron was trained from scratch. The WaveGlow's official implementation code was not changed, however the model was fine-tuned using approximately three hours of North Sámi audio with a single speaker.

With training on a single GPU, fine-tuning of both Tacotron models and the WaveGlow model took 1 month. Training of the ForwardTacotron took around three days.

6.2 ASR model

The speech recognition model used for this project is the Wav2Vec model which is a part of the publicly available Huggingface library and the Fairseq package. It was introduced in autumn 2020 by Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, Michael Auli and was the most recent open-sourced and state-of-the-art model for ASR available at the time of working on this paper (Baevski et al. 2020).

For training Wav2Vec independently from the TTS model, the HuggingFace API could be used. However, to implement the connection of two tasks (speech recognition and speech synthesis), our project required custom behaviour of the Trainer class for Transformers. To achieve this we implemented the training functions by ourselves to override the default behaviour and allow for custom training.

As for the data, we now downsample our audio to 16kHz (as it is what the original base Wav2Vec model was trained on). In addition, since the speech recognition task can benefit from training on multiple voices, we can now increase the amount of training data by mixing our female and male recordings, having a train/test split with a ratio of 3700/300.

It needs to be mentioned that our usage of the Wav2Vec model does not rely on the language model itself, but only on the audio representation of the target language. We were aware of the possibly poorer performance of the ASR model without the language model. For example, without bound LM for ASR it is more difficult for the ASR model to learn to write naturally, not knowing which words are meaningful in the sentence. Considering the time constraints of this project, we did not consider implementing the LM for now, but that could be a possible extension of this project.

6.3 Dual transformation

The dual transformation is part of the training class for TTS and ASR models. This part of the training requires an additional set of data—unpaired data—only speech and only text. No specific preprocessing is required for unpaired data, apart from downsampling the audio and splitting text in an array of sentences.

The dual transformation is a part of the regular training loop, however, it combines supervised and unsupervised training (see Figure 3).

First, we are going through the forward pass of the TTS model on our regular (paired) training data. We, as usual, calculate the loss value and store it. Similarly, we run the forward pass with the ASR model on the paired data, calculate and store the loss value.

Now we are in the second phase of our training loop—unsupervised training. In this way, unpaired text (sentences) are becoming an input to our TTS model which is already able to generate some kind of speech (inference step with the TTS model). This speech together with input sentences is now stored as a temporary corpus for training our ASR model. The loss value of the ASR model is being calculated and stored.

Similarly, unpaired speech samples are given to our ASR model to run inference. The produced transcriptions and input speech samples are then stored as a temporary corpus for the training of our text-to-speech model. The loss value of the TTS model is being calculated and stored.

At the final part of each training step, we are combining all the loss values, calculating gradient with derivative of loss and updating the models' parameters.

Such training is designed to run on each epoch, constantly updating temporary ASR and TTS corpora and letting both models learn from new unpaired data on the fly. With this training loop, enriched with the dual transformation functionality we 1) repeat a part of Microsoft's low resource project experiment, 2) increase the amount of training data for both ASR and TTS and 3) let both models 'talk to each other'.

Parameter	TTS model	ASR model
Steps	600 000	30 000
Learning rate	1e-4 for 300 000 steps, 2e-5 for 300 000 steps	1e-4
Batch side	32 for 300 000 steps, 16 for 300 000 steps	8

Table 2. Training hyperparameters

We use default training parameters to train ForwardTacotron and Wav2Vec. As our TTS model (ForwardTacotron) requires longer training time, we are using adaptive learning rate, as recommended by its authors (see Table 2).

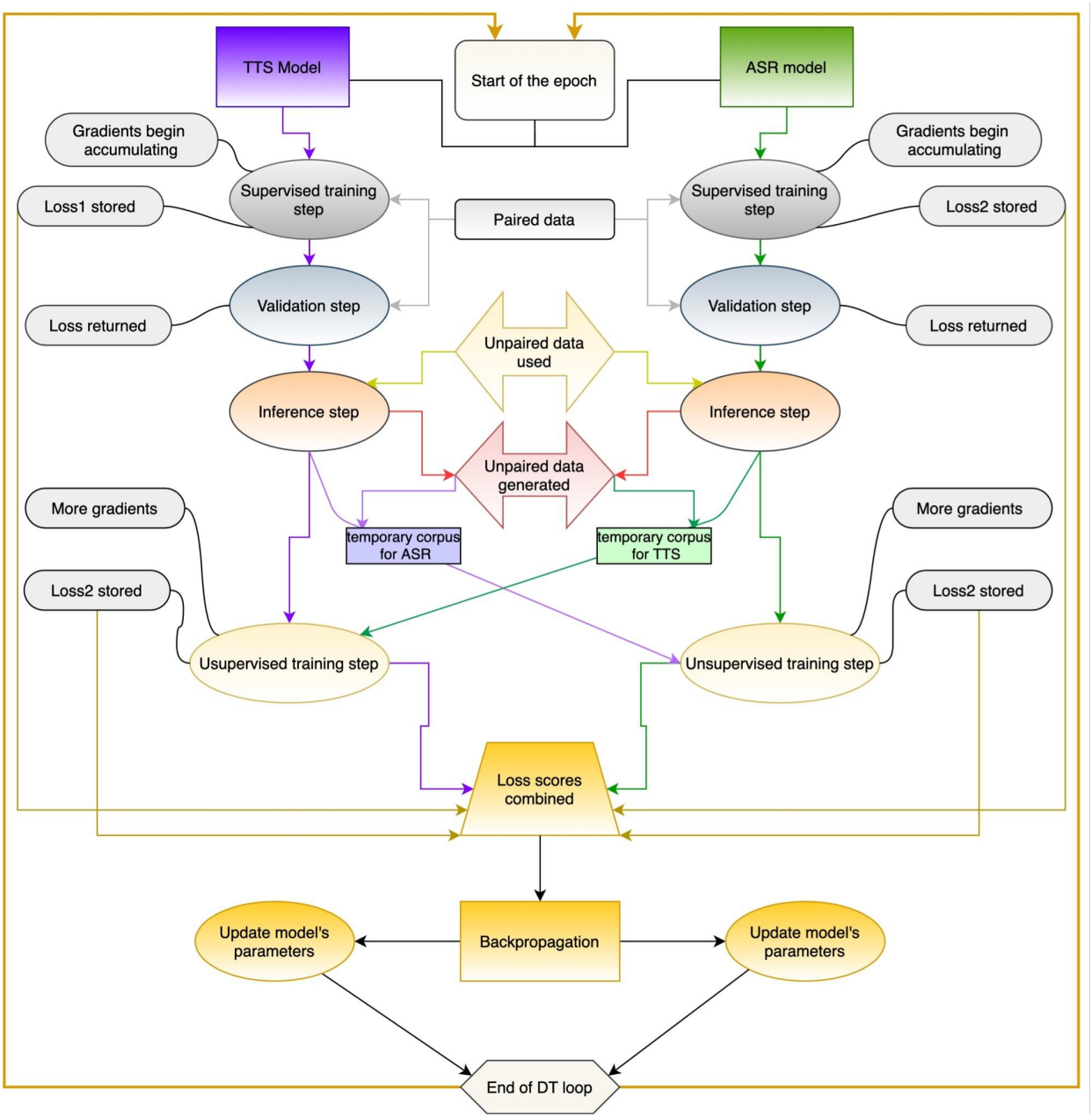


Figure 3. Dual transformation loop architecture.

Since the description of the dual transformation concept (or similar) was mentioned before only in three papers, two of which were published from the work done at Microsoft (no open-source examples are available), we were required to make some assumptions to implement our system.

For example, it was not clear to us whether the DT loop should concern the training from scratch or resuming from a somewhat fine-tuned checkpoint, when both approaches make sense. Following the Xu et al. (2020) work, we decided to run the DT loop on somewhat trained, but not converged models.

On a single 11 GB GPU (that was used during most of the time of the work on this project) the task of dual transformation appeared to be infeasible. Even with data loading optimizations that were made, the machine learning task was still not successful on a 11 GB GPU and would regularly run out of memory

We used the PyTorch library for the machine learning task, which gives an option to run some of the operations on CPU and reserve GPU only for the task of accumulating gradients for each of our models, by interchanging their locations back and forth between CPU and GPU. However, since we need not just

to train two models one after another, but want them to share some information between each other, the backpropagation that we make to perform a learning step should happen simultaneously. The backwards move in PyTorch happens quietly, and cannot be directly assigned a location (CPU or GPU) by us. The gradients produced and accumulated during the training step are being calculated during the backpropagation pass, which would inevitably happen on the same node as where those gradients were created. Therefore, if we make a training step on the GPU, we cannot backpropagate onto the CPU and vice versa, even if we have declared some moves of the models between devices.

Since the gradient accumulation and calculation are the most important and the most time-consuming processes during the ML training loop, we were not able to perform a meaningful (long enough) dual transformation training on CPU only to receive at least some results for evaluation.

When looking for solutions for the out-of-memory issue, multi-GPU training was researched, but we had insufficient resource budget to attempt it.

Thanks to Divvun, on the final stage of this project, we received access to a "supercomputer", the Norwegian academic high-performance computing and storage service (Sigma2/Metacenter, n.d.). With 16GB GPU (which is not a high-computing power GPU nowadays[2]) and with a declared size of the data (around 4000 training pairs for ASR and TTS) we managed to test our implementation and run the training.

[2] GPU benchmark: https://benchmarks.ul.com/compare/best-gpus

7. Results

We have chosen the loss value as the metric to monitor during the training of both models, and the performance of the models at the final stages was evaluated by one North Sámi native speaker. Apart from this, to decide whether the models are progressing, we would randomly run inference with the model to evaluate the noise level of output and would always keep track of mel and alignments plots.

The most common metric used for evaluating speech synthesis systems is the MOS (mean opinion score) scale, which requires independent listeners to evaluate the speech that they listen to in terms of sound quality, listening effort, comprehension problems, articulation, pronunciation, speaking rate and pleasantness. This kind of evaluation was not performed for this project, as we didn't have a dedicated focus group of native speakers to evaluate the results of this project. For evaluation of the speech recognition system, apart from the loss curve, we were also observing the word error rate (WER) score.

During the training of Tacotron2 we reached a reasonably good alignment level, however, the speech was not high quality yet. The output was noisy, with a somewhat robotic voice, it would contain pauses or lack full words. Apart from that, a few sounds were not learned consistently. We noticed, for example, that the letter *č* that has a sound /ʧ/ would be pronounced like /s/ in the beginning of the word, but /ʧ/ if in the middle. More training did not fix this problem, so we migrated to training with a reduction factor (as explained in Section 6) and after stabilizing the alignments (the alignments of the ground truth and the generated data were looking very similar to each other), we finalised our training with Dual Transformation and ForwardTacotron.

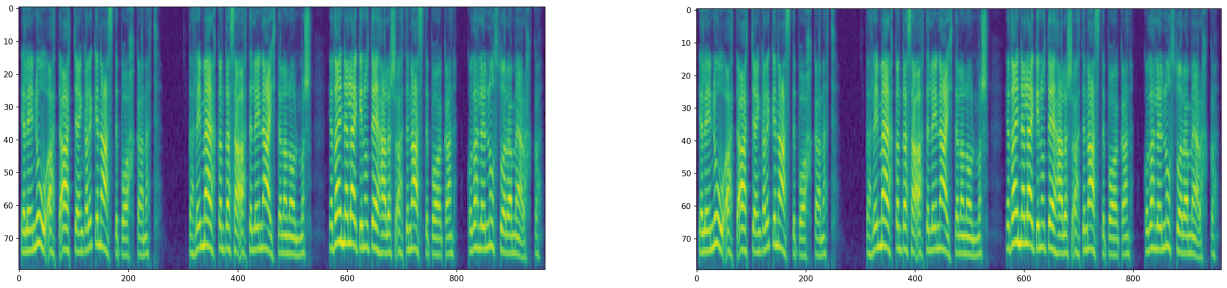


Figure 4. Ground truth mel-spectrogram (on the left) and ground mel-spectrogram (on the right) look almost identical when training with the DT loop.

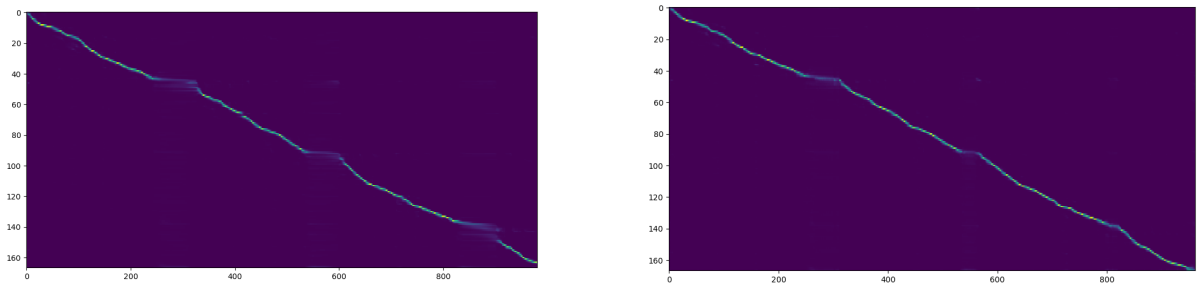


Figure 5. Character-phoneme alignments during training. Left—generated, right—ground truth alignment. The generated alignments still have areas (blurrier parts of the plot) which indicate that some character might be mispronounced.

At the final stage of our training, we could see that our mel and alignments plots look good (e.g. diagonal for alignments). As the Griffin-Lim produced outputs sound not natural, for demonstration purposes we smooth it out with the fine-tuned Wave-Glow vocoder, which gives us a much more natural sounding of the speech synthesis model. In addition, our native speaker concluded that the speech quality is overall good.

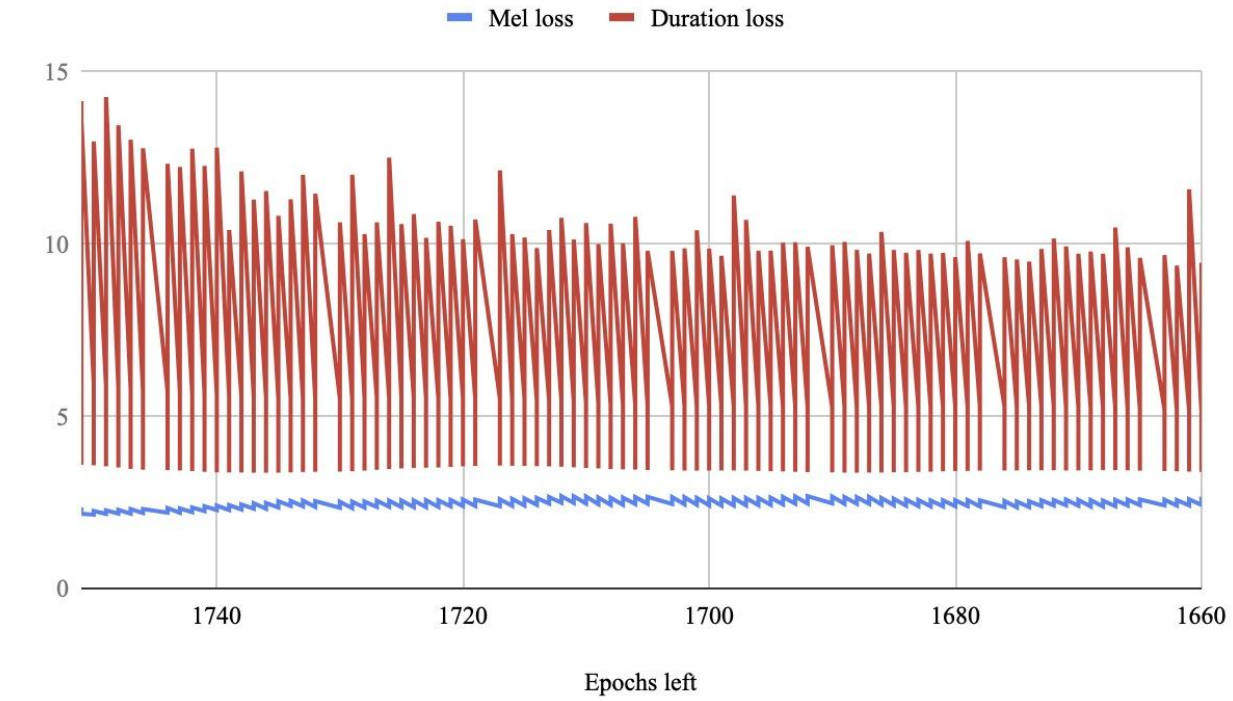
As for our ASR model, one type of error of the text outputs was expected—missed or extra letters in words. The way the ASR misspells the words in our case is the same as described in Facebook's paper. Since we did not use the language model for predicting the output, our predictions are spelled phonetically. From Table 3 we can see that it is sometimes difficult for a model to know when a double letter should be used and when a single one (since we are working with a strongly agglutinative language, this could be learned from the LM), when the word is finished and when a white space should be put, etc. At the same time, from the table, we can see that even in the misspelled words, the mistakes that are present are not very damaging for the text understanding. In addition, the erroneous output like in Table 3 is very easy to handle for Divvun's spell checking software, to guarantee 0% WER score on inference.

As we did not train our models from scratch with the dual transformation, we had an opportunity to also compare how different they behave in supervised and unsupervised settings. Before our DT training, our ASR model (after short supervised training) was at 58% WER and 0.5 loss. We trained it only for around 800 steps. Our TTS model was already trained for 400 000 steps and was at 2 and 6 for duration and mel loss respectively.

Predicted	Actual word
nugo	nu go
áiggit	áiggiid
gohl	goal
olmmošlei	olmmoš lei
rinškit	rinškkit
biebmo binná	biebmobinná
earáláhkái	earáláhkái

Table 3. Some examples of the most common types of errors when inferencing with the ASR model.

In the first several dozen loops of the dual transformation the loss scores of the models start to drastically increase. We now can see that ASR loss is at almost 4 and TTS losses are at 3 (mel) and the duration loss fluctuates between the values of 3 and 14 (see Figure 6 & 7). After a rapid jump, the losses start to go down very slowly, around 10 times slower than in the supervised setting. This behaviour is expected, as the DT training time is supposed to be much longer as our models now receive new data and exchange the knowledge between each other. On the other hand, we also can see that our WER score for the ASR model continues to be going down, even when dealing with unseen data, falls and reaches 41%.



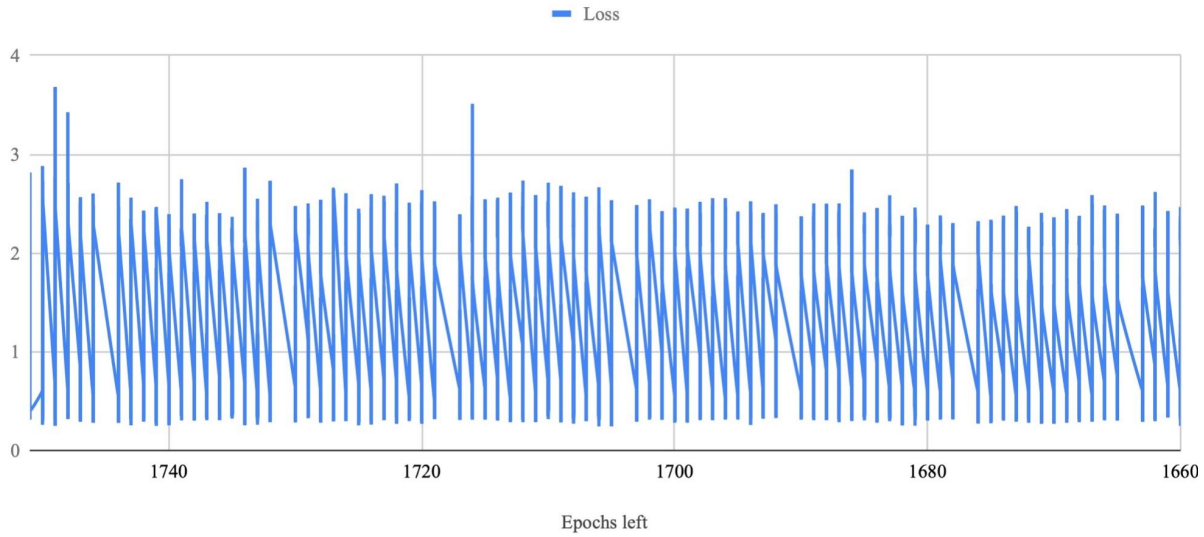


Figure 6 & 7. A snippet of training dynamics at the beginning of training with the dual transformation loop. Top—graph shows the loss curve while training the TTS model, bottom—ASR model. High peaks refer to unsupervised training phase, and low peaks—supervised. The curve straightens very slowly and the high peaks become lower.

8. Conclusions and further work

In this project we have demonstrated that by leveraging open-source resources we can 1) develop a functional technology for low-resource languages with little data available, 2) use transfer learning to boost the performance and fill in the gaps in data, and 3) make a dual connection between the two tasks to increase the amount of data and achieve more robust models in the end.

As for the possible extension of the experiment, a few points should be considered for the future work:

1. This work has shown that using knowledge learned from resource-rich languages gives a good warm-start base for training on the target (low-resource) language. Our motivation of choice of the base language (English for pretraining in our case) was made from the point of being limited by time, GPU resources and choosing only from publicly available models. However, it is important to consider the *language* aspect, and in case of having more resources, repeat the experiments with another base language—a language that would be closer and have more in common with the target language. For North Sámi, it would be good to use the Finnish language as the base language. Being a bit closer in the language group than English, pretraining on Finnish might give better results at inference and possibly would require shorter time to fine-tune and converge. During the timeline of this project, it was not possible to conduct this experiment mainly because we did not have access to a Finnish language dataset of similar size and scope as there are available for English models.

2. The current implementation of the project is working stably and in essence the complex training structure of the TTS model (Tacotron2-Tacotron1-ForwardTacotron-WaveGlow) would not be detrimental to the efficiency of the model (for example, it does not influence the inference time therefore would not be detrimental to the final product of the TTS system). However, it is a good practice to make sure that our implementation is as concise as possible and if we can avoid one of the middle steps during the training, we should do so. Our experiments have shown that Tacotron2 itself is not a good base model in our case, so we migrated the training to Tacotron1. It's important to repeat the experiment only with Tacotron1 and if the assumption that we do not need Tacotron2 training at all will be confirmed, we would be able to remove one item from the chain of training.

3. Most probably, the neural vocoder, as an item from the architecture chain, cannot be removed because it is difficult to rely on the Griffin-Lim algorithm. However, more fine-tuning experiments should be conducted with different neural vocoders (CNN and GAN based) to establish which one would be better for our task.

4. As for the ASR model, we are convinced that at this time there is no better base model for low-resource languages than Facebook's XLSR-Wav2Vec2. As it was already mentioned, training the base model from scratch on, for example, the Finnish language, would be a good experiment. In addition, we recommend extending the ASR model with the language model that will hopefully shrink the WER score even more. It is worth an effort to try to reach a WER score below 30% to reach at least the bottom level of the current industry-level performance for ASR (Leaderboard, n.d.). Apart from that, we do not really see any point of experimenting with other approaches for the ASR models, given the very good performance of Wav2Vec2 already in this project.

5. We believe that the Dual Transformation loop may greatly benefit parallel training in the low-resource settings. Being limited by the amount of data, we put the biggest focus on implementing the dual connection between the two models. We saw some results when training the TTS and ASR models separately, without the DT. They have not converged after the aggregate of a session of 3 weeks training. Our DT training, however, had run only for one week after what we already reached subjectively good results at inference. Therefore, one of the tasks

for the future development of this project would be bringing the models to convergence and giving a report of scores differences when training with and without DT. Even when we can subjectively report slower but more stable learning progress in case of DT, it is very important to be able to numerically evaluate the difference in performance with respect to the time spent on training and scientifically prove the benefit of parallel training.

We consider the present results of our experiment to be very promising and worthy of continued development. In the future, we will keep working on the improvement of the scores of our ASR and TTS models as well as on the further adaptation of our models for web and mobile applications.

References

Apple. (n.d.). iOS and iPadOS - Feature Availability. Retrieved May 8, 2021, from https://www.apple.com/ios/feature-availability/#siri

Baevski, Alexei & Zhou, Henry & Auli, Michael. (2020). wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations.

Berlitz. (n.d.). What is the Berlitz Method. Retrieved May 12, 2021, from https://www.berlitz.com/en-dk/about-berlitz/berlitz-method#:%7E:text=The%20Berlitz%20Method%20lets%20you,where%20the%20language%20is%20spoken.

Chomsky, N. (1988). *Language and the problems of knowledge*. MIT Press. p. 24.

Cooper, Erica (2019). Text-to-Speech Synthesis Using Found Data for Low-Resource Languages. http://www.cs.columbia.edu/speech/ThesisFiles/erica_cooper.pdf

de Korte, Marcel & Kim, Jaebok & Klabbers, Esther. (2020). Efficient neural speech synthesis for low-resource languages through multilingual modeling. https://arxiv.org/pdf/2008.09659.pdf

Dickson, B. (2020, April 17). AI Could Save the World, If It Doesn't Ruin the Environment First. PCMag UK. https://uk.pcmag.com/artificial-intelligence-2/125690/ai-could-save-the-world-if-it-doesnt-ruin-the-environment-first

Gold, Ben, Morgan, Nelson, Ellis, Dan. (2011). Speech and Audio Signal Processing: Processing and Perception of Speech and Music. John Wiley & Sons. ISBN 1118142918, pages 9–13

Google Cloud. (n.d.) Language support. Cloud Speech-to-Text Documentation. Retrieved May 8, 2021, from https://cloud.google.com/speech-to-text/docs/languages

Griffin, Daniel & Lim, Jae. (1983). Signal estimation from modified short-time Fourier transform. IEEE Transactions on Acoustics, Speech, and Signal Processing. ASSP-32. 804 - 807. 10.1109/ICASSP.1983.1172092.

Hannun, A.Y., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., Prenger, R., Satheesh, S., Sengupta, S., Coates, A., and Ng, A. (2014). Deep Speech: Scaling up end-to-end speech recognition. ArXiv, abs/1412.5567.

Ito, Keith (2017). The LJ Speech dataset. https://keithito.com/LJ-Speech-Dataset/

Karita, S., Chen, N., Hayashi, T., Hori, T., Inaguma, H., Jiang, Z., Someki, M., Yalta, N., Yamamoto, R., Wang, X., Watanabe, S., Yoshimura, T., & Zhang, W. (2019). A Comparative Study on Transformer vs RNN in Speech Applications. 2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU), 449-456.

Klatt, D (1987). "Review of text-to-speech conversion for English". Journal of the Acoustical Society of America. 82 (3): 737–93. Bibcode:1987ASAJ...82..737K. doi:10.1121/1.395275. PMID 2958525

Leaderboard. (n.d.). TIMIT Benchmark (Speech Recognition). Retrieved May 19, 2021, from https://paperswithcode.com/sota/speech-recognition-on-timit

Liu, Peng & Wu, Xixin & Kang, Shiyin & Li, Guangzhi & Su, Dan & Yu, Dong. (2019). Maximizing Mutual Information for Tacotron.

Molloy, B. D. (2021, January 24). The great graphics card shortage of 2020 (and 2021). BBC News. https://www.bbc.com/news/technology-55755820

Panayotov, V., Chen, G., Povey, D. and Khudanpur, S. (2015). "Librispeech: An ASR corpus based on public domain audio books," IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2015, pp. 5206-5210, doi: 10.1109/ICASSP.2015.7178964.

Prenger, Ryan & Valle, Rafael and Catanzaro, Bryan. (2019). Waveglow: A Flow-based Generative Network for Speech Synthesis. 3617-3621. 10.1109/ICASSP.2019.8683143.

PyTorch. (n.d.). Transfer Learning for Computer Vision Tutorial — PyTorch Tutorials 1.8.1+cu102 documentation. Retrieved May 12, 2021, from https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html

Rehm Georg. (2020). Research for CULT Committee – The use of Artificial Intelligence in the Audiovisual Sector, European Parliament, Policy Department for Structural and Cohesion Policies, Brussels

Ren, Yi & Ruan, Yangjun & Tan, Xu & Qin, Tao & Zhao, Sheng & Zhao, Zhou & Liu, Tie-Yan. (2019). FastSpeech: Fast, Robust and Controllable Text to Speech.

Ren, Yi, Tan, Xu, Qin, Tao, Zhao, Sheng, Zhao, Zhou and Liu, Tie-Yan. (2019). Almost Unsupervised Text to Speech and Automatic Speech Recognition.

Schäfer, C. (n.d.). as-ideas/ForwardTacotron. GitHub. Retrieved May 12, 2021, from https://github.com/as-ideas/ForwardTacotron

Schwab, J. F., & Lew-Williams, C. (2016). Repetition across successive sentences facilitates young children's word learning. *Developmental psychology*, *52*(6), 879–886. https://doi.org/10.1037/dev0000125

Sciforce. (2020, February 13). Text-to-Speech Synthesis: an Overview. Medium. Retrieved May 8, 2021, from https://medium.com/sciforce/text-to-speech-synthesis-an-overview-641c18fcd35f

Shen, Jonathan, Pang, Ruoming, Weiss, Ron, Schuster, Mike, Jaitly, Navdeep, Yang, Zongheng, Chen, Zhifeng, Zhang, Yu, Wang, Yuxuan, Skerrv-Ryan, Rj, Saurous, Rif, Agiomvrgiannakis, Yannis, Wu, Yonghui. (2018). Natural TTS Synthesis by Conditioning Wavenet on MEL Spectrogram Predictions. 4779-4783. 10.1109/ICASSP.2018.8461368.

Sigma2/Metacenter. (n.d.). The Norwegian academic high-performance computing and storage services. Retrieved May 8, 2021, from https://documentation.sigma2.no/index.html#

Strubell, Emma & Ganesh, Ananya & Mccallum, Andrew. (2019). Energy and Policy Considerations for Deep Learning in NLP. 3645-3650. 10.18653/v1/P19-1355.

Taigman, Y., Wolf, L., Polyak, A., and Nachmani, E. (2018). VoiceLoop: Voice Fitting and Synthesis via a Phonological Loop. arXiv: Learning.

Taylor, P. (2009). Text-to-Speech Synthesis. Retrieved May, 8, 2021, from https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.118.5905&rep=rep1&type=pdf

Tjandra, Andros & Sakti, Sakriani & Nakamura, Satoshi. (2017). Listening while speaking: Speech chain by deep learning. 301-308. 10.1109/ASRU.2017.8268950.

Van den Oord, Aaron; Dieleman, Sander; Zen, Heiga; Simonyan, Karen; Vinyals, Oriol; Graves, Alex; Kalchbrenner, Nal; Senior, Andrew; Kavukcuoglu, Koray (2016-09-12). WaveNet: A Generative Model for Raw Audio. 1609. arXiv:1609.03499. Bibcode:2016arXiv160903499V.

Wang, Y., Skerry-Ryan, R., Stanton, D., Wu, Y., Weiss, R.J., Jaitly, N., Yang, Z., Xiao, Y., Chen, Z., Bengio, S., Le, Q.V., Agiomyrgiannakis, Y., Clark, R., & Saurous, R. (2017). Tacotron: Towards End-to-End Speech Synthesis. INTERSPEECH.

Xu, Jin & Tan, Xu & Ren, Yi & Qin, Tao & Li, Jian & Zhao, Sheng & Liu, Tie-Yan. (2020). LRSpeech: Extremely Low-Resource Speech Synthesis and Recognition. 2802-2812. 10.1145/3394486.3403331.