Engineering Applications of Artificial Intelligence 136 (2024) 108998

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

An embedded end-to-end voice assistant

Luca Lazzaroni *, Francesco Bellotti, Riccardo Berta

Department of Naval, Electrical, Electronic, Telecommunications Engineering (DITEN), University of Genoa, Genoa, Italy

ARTICLE INFO

Keywords:
Voice assistants
Smart vehicles
Edge computing
Embedded systems
Transfer learning
Automatic speech recognition

ABSTRACT

Voice assistants are spreading in various environments, such as houses and cars, bringing the possibility of controlling heterogeneous Internet of Things devices with simple voice commands. However, massive use of the cloud connection for speech processing requires an efficient and robust Internet connection and raises concerns in terms of privacy. Therefore, we propose an end-to-end solution able to work totally offline, based on a system architecture combining different Deep Learning models to implement all the steps of the speech elaboration process. Being interested in targeting the Italian language, we exploited the transfer learning paradigm, which allows leveraging models trained in English on large datasets and fine-tuning them to the target language on a smaller dataset. The proposed system architecture is configurable and easily extensible to other languages. Experimental results in an automotive application use case show that our solution outperforms the other embedded models and achieves performance comparable to state-of-the-art cloud-connected solutions for Automatic Speech Recognition. Moreover, overall latency is significantly reduced by eliminating the need to connect to the cloud.

1. Introduction

Connected and automated vehicles are becoming Internet of Things (IoT) -rich environments, providing a wide set of functionalities and services to drivers and passengers (Son et al., 2022). Examples of such vehicular systems include infotainment and heating, ventilation, and air conditioning (HVAC) (e.g. (Kazmi et al., 2019; Massai et al., 2019),), but also advanced driving assistance systems (ADAS) and automated driving functions (ADF) (e.g. (Häfner et al., 2022),), fuel/energy management (e.g. (Rhode et al., 2020)), predictive maintenance systems (e.g. (Carvalho et al., 2019),), and vehicular data querying (Andrade et al., 2023). The need for comfortably managing such systems is a key requirement on the part of drivers and passengers. Various Advanced Human-Machine Interaction (HMI) solutions are being investigated and developed to face such a challenge (e.g. (Li et al., 2024; Massai et al., 2019; Okumura, 2019),).

Voice Assistants (VAs) are acquiring ever more relevance in this context. Industry giants such as Amazon, Google, Microsoft, and Apple propose Alexa, Assistant, Cortana, and Siri, respectively. The major benefit of VAs is the possibility of controlling heterogeneous IoT devices with voice commands, also in smart vehicles. However, privacy and security are serious concerns, especially with cloud-based solutions (Dos Anjos et al., 2020; Hernández Acosta and Reinhardt, 2022; Hoy, 2018; Seymour et al., 2023). While waiting for the wake-word that prompts a VA's dialogue management system, smart speakers are always listening. Then, once awoken, a VA begins recording audio clips of what the user says. Thus, a stream of clips is continuously transmitted to a cloud server, where the audio is processed to understand the user's speech and formulate a response. This exposes user data and makes the system vulnerable to cyber-attacks. Moreover, the data processing requires an efficient and stable Internet connection. A real-time response is also needed, otherwise there is a risk of driver frustration and even safety issues. Such requirements, which are key for the automotive environment, suggest the value of offline VAs to support privacy, increase robustness, and guarantee availability in every context (Liu et al., 2023). Some online VAs, such as Apple Siri and Google Assistant, offer also some offline voice assisting capabilities, such as for setting alarms, sending text messages, or playing music. However, the range of supported functionalities is very limited compared to the fully online operability. Fully offline VAs, on the other hand, are less widespread and offer fewer capabilities.

The edge computing paradigm, performing the computation locally, could be useful in addressing these points. In addition to avoiding (or reducing) the need for transferring private information, edge devices bring benefits in terms of latency, energy consumption, and bandwidth occupation (Berta et al., 2022; Shi et al., 2016; Wang et al., 2019).

* Corresponding author.
E-mail addresses: luca.lazzaroni@elios.unige.it, luca.lazzaroni@edu.unige.it (L. Lazzaroni).

https://doi.org/10.1016/j.engappai.2024.108998
Received 26 May 2022; Received in revised form 29 June 2024; Accepted 14 July 2024
Available online 19 July 2024

Cloud-based solutions can count on potentially much larger computational and storage resources, which allows for achieving high performance and versatility. However, considering the need for an embedded system to handle and process data locally, a great opportunity is provided by new generation edge devices, that feature impressive computational power, also with GPUs (e.g. (Franklin, 2018),). Moreover, Machine Learning (ML) models can be trained on large datasets in the cloud and then deployed on embedded devices thanks to optimized Deep Learning (DL) techniques (Mittal, 2019). This promises to build offline VAs able to achieve performance levels comparable to cloud-connected solutions, also in terms of the variety of processable commands.

The state of the art already presents AI-based solutions in this direction (Polyakov et al., 2018). have developed an ML-based VA able to work offline but based on low-performance models such as pocketsphinx, which achieves a word error rate (WER) of 31% in automatic speech recognition (ASR) of the English language. Kaldi (Povey et al., 2011) and Picovoice Leopard (Picovoice, 2022a) are offline open-domain speech-to-text engines already able to achieve results comparable to the state-of-the-art in terms of WER (Brinckhaus et al., 2021). The three above tools process English speech, with Picovoice being able to work also in French, German, and Spanish. However, the literature still lacks presentations and performance analyses of end-to-end offline systems.

The goal of the work presented in this article is to investigate feasibility of an offline VA embedded system able to achieve state-of-the-art performance (i.e., performance of cloud-based systems) in terms of speech recognition, extensibility to different languages, and versatility (i.e., being trained on large and generic voice datasets). Versatility is important given the rapid and large evolution of the services and functionalities available in a vehicle. We are interested in a non-mainstream language (i.e., neither English nor Chinese), such as Italian. Such languages lack the huge voice datasets that are needed to effectively train a DL ASR model from scratch. However, they represent the vast majority of potential users in the world, and the transfer learning (TL) approach (from English to Russian, Spanish, and German (Huang et al., 2020)) has already shown positive results.

We are interested in developing a VA system architecture on a high-end embedded device, building on publicly available AI resources. We investigate the needed modules, their interconnections, and performance. We analyze the relevant ML models, training modalities, and datasets.

We emphasize three key novel contributions. First, we strive to achieve offline performance comparable to cloud-based systems, which is a significant advancement in the field. Second, we focus on supporting non-mainstream languages like Italian, recognizing their importance in serving a vast majority of potential users worldwide. Finally, we undertake a comprehensive analysis of system modules, ML models, and training methodologies, providing insights and advancing the current understanding in the domain of offline VAs.

Table 1
Macro-use-cases of the car VA.

Subsystem	Examples
Media	- Stop the track - Fast forward the track by 15 s
Phone	- Please, call John - Call mum via WhatsApp
Heating, Ventilation, and Air Conditioning (HVAC)	- Set the temperature to 22° - Turn off the heating of the front left seat
Vehicle state	- How many liters are consumed per 100 km? - Tell me the tire pressure
Drive mode	- Select sport mode - Enter eco mode
Location-based services (offline)	- Take me to Rome - Find the nearest Italian restaurant

To give an idea of the targeted scope of the system, Table 1 presents a list of voice command examples, grouped by their relevant subsystem. While the dialogue management system is designed to work completely offline, it is of course able to manage also commands whose execution requires an Internet connection (e.g., the download of a song or a roadmap, the interaction with an online navigator, or weather information system). To this end, also intents requiring an internet connection have been considered in the system development.

The remainder of the article is organized as follows. Section 2 gives an overview of the state of the art. Section 3 presents the system architecture of the proposed VA system, while Section 4 details its implementation. Experimental results are provided in Section 5, and Section 6 draws conclusions and indicates possible directions for future research.

2. Related work

This section briefly presents related works in speech processing. We mainly focused on two languages, Italian and English. The former is the target language of our system, the latter is the reference language in the field in terms of scientific literature contributions and commercial product availability.

2.1. Datasets

For training our VA we investigated open-source AI resources, due to the cost of proprietary datasets, in the order of the tens of thousands of dollars. Italian open-source datasets are few, with only a small amount of speech hours available. An emerging solution is represented by Mozilla Common Voice, a crowd-sourced, open-source, multi-language dataset (Ardila et al., 2020). Using the Common Voice website (Mozilla, 2020), people can record their voices by reading sample sentences proposed by the website. Recordings are then verified by other contributors using a voting system. Each record is assessed by three users in a poll, and if it collects two or more votes in favor, it is marked as valid and added to the dataset (training, validation, or test set), otherwise is considered invalid and discarded. Those clips which did not receive enough votes to be marked as valid or invalid are released as “other”. The Italian version is currently composed of a total of 310 validated hours (MP3 format, mono, sample rate 48 kHz). The dataset has manifest (in TSV format) files for training, validation, and test phases, and other secondary categories including an “others” one. Manifests contain, for each audio clip, the transcription of the sentence uttered.

M-AILABS is a free-of-charge, freely useable multi-language dataset (Solak, 2019). Data are retrieved from LibriVox (McGuire, 2014) and Project Gutenberg (Hart, 1971) and divided by speaker name and sex. This makes this dataset suited for tasks where a single speaker is needed, i.e., Speech Synthesis. The Italian branch of the dataset consists of two voices: male and female. The largest portion is occupied by the male voice, with a total of 18 h of speech. All the clips are recorded at a 16 kHz sample rate and provided in WAV format, mono. The read texts were published between 1884 and 1964 and refer to literature classics, so the pronunciation is sometimes courtly and emphatic. Manifest files are provided in CSV and JSON formats. Facebook AI released Multilingual LibriSpeech (MLS) (Pratap et al., 2020), an open-source dataset in eight different languages, including Italian. Like M-AILABS, this dataset leverages audiobooks from LibriVox (McGuire, 2014), but it is slightly larger, including, for the Italian language, 28 different speakers totaling 279 h of speech.

(H. Liu et al., 2023) have recently proposed LogiQA 2.0, a large-scale logical reasoning reading comprehension dataset. It supports a two-way natural language inference (NLI) task, which offers greater generalizability (thus facilitating downstream tasks) than the question answering task.

2.2. Models and toolkits

Speech models are typically built, exploiting such datasets, through speech toolkits. Recently, many Italian language-enabled toolkits have been developed, some of which also provide pre-trained models, that can be fine-tuned in some specific applications.

2.2.1. Speech recognition

Vosk (Alpha Cephei, 2020) is a speech recognition toolkit characterized by a large vocabulary transcription, reconfigurable vocabulary, and ease of use and installation. An Italian model of reduced dimensions is available, which is suitable for use on Raspberry Pi or Android devices. Mozilla DeepSpeech (Mozilla, 2022) relies on the Common Voice Italian dataset (Ardila et al., 2020), one of the largest open-source datasets, for performing Automatic Speech Recognition (ASR), and has been developed using end-to-end Deep Learning (DL). The system core is a Recurrent Neural Network (RNN) trained to ingest speech spectrograms and generate text transcriptions. Facebook AI Research released wav2-letter (Collobert et al., 2016), an ASR tool that also provides pre-trained models, including an Italian one derived from Multilingual LibriSpeech [6]. NVIDIA's toolkit for ASR is NeMo (Kuchaiev et al., 2019), which includes pre-trained models as well. In addition to ASR, NeMo also supports Voice Activity Detection (VAD), Keyword Spotting (KWS), and Text to Speech (TTS) all in one package, so that it can perform the entire conversational process. A comparison among the above-described tools is performed in (Nagari, 2020), which shows a slight superiority of the NVIDIA solution. Another DL-based speech recognition system is presented in (Weng et al., 2023). The authors propose a semantic communication model that extracts text-related semantic features from the input speech using a Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN)-based encoder. The received semantic features are then converted to text information through a decoder. Then, speech synthesis is performed to re-generate the speech signals. Wav2-vec 2.0 (Baevski et al., 2020) is a framework for self-supervised learning of speech representations. It involves masking the latent representations of the raw waveform and addressing a contrastive task using quantized speech representations. The achieved WER is equal to 4.8/8.2 on test-clean/other in the Librispeech dataset (Panayotov et al., 2015).

Since Italian datasets' dimensions are modest compared to English ones, especially when multiple-speaker datasets are considered, an idea is to exploit TL (Torrey and Shavlik, 2010), which has been recently used to compensate for the limited speech hour availability of most of the natural languages, apart from English and Chinese (Wang and Zheng, 2015). Mozilla DeepSpeech provides a TL version fine-tuned from the Common Voice English dataset to some other languages, including Italian. An early research application of fine-tuning is presented in (Huang et al., 2020), where the NeMo toolkit is used to perform TL from English to Spanish and Russian. In particular, the QuartzNet 15x5 model (Kriman et al., 2020) is fine-tuned with the Common Voice Spanish and Russian datasets. Results show that both (i.e., Spanish and Russian) fine-tuned models outperform their trained-from-scratch counterparts. The main hurdle lies in the computing resources requested for the training phase. This aspect is dealt with in (Huang et al., 2019), which provides specifications for proper training, along with the results achieved by the QuartzNet model with the NeMo framework.

Adversarial techniques are being explored for improving the training of ASR neural systems, for instance to support dysarthric and elderly speech (Jin et al., 2024).

2.2.2. Natural language understanding

After the speech recognition step, natural language understanding (NLU) is needed to infer the meaning from the text provided by the ASR model. Besides well-established solutions, that require an Internet connection (e.g., Google Dialogflow (Sabharwal and Agrawal, 2020), Amazon Lex (Williams, 2018), Facebook wit.ai (Mitrevski, 2018), and Microsoft Bot Framework (Biswas, 2018)), Rasa (Bocklisch et al., 2017) is an offline open-source python module to make Machine Learning (ML)-based dialogue management and language understanding. It also offers the possibility of using pre-defined pipelines, including the spaCy one (Vasiliev, 2020). SpaCy is an open-source Natural Language Processing (NLP) tool offering pre-trained models in 64 different languages. SpaCy is a powerful tool for a multi-language VA system and, supported by Rasa, it allows to perform NLU completely offline. A default pipeline is also available, which allows the user to train a model from scratch. This is particularly suited for domain-specific applications. DeepPavlov (Burtsev et al., 2018) is an open-source library for the development of dialogue agents in Python. It features three models, namely intent classification, entity recognition, and spelling correction. Similarly to Rasa, the models to be included in the training can be defined through a pipeline (but it is less customizable than the Rasa one). Recently, also ASR and NLU end-to-end solutions have been proposed. For instance (Saxon et al., 2021), offers an English, transformer-based (Vaswani et al., 2017) model for Spoken Language Understanding (SLU). The model (115M parameters) achieves a 99% accuracy on the Fluent Speech Commands dataset (Lugosch et al., 2019). Snips (Coucke et al., 2018) is an SLU platform dedicated to IoT microprocessors. It can work offline with low resource consumption. However, it has recently been acquired by a private company, therefore the code is not publicly documented and is no longer open-source. Massai et al. (2019) have proposed Paval, a virtual personal assistant that suggests local points of interest (POIs) and services by analyzing users' natural language queries. The authors show that, in this domain, the assistant results are better ranked than the state-of-the-art general-purpose systems, such as Google Assistant, Apple Siri, and Microsoft Cortana. Besides NLP, the Paval system exploits semantic technologies and external knowledge to retrieve geo-located data. Our focus, instead, is on an embedded VA able to process any type of query in a smart vehicle environment (e.g., Table 1). The VA could then be improved by adding semantic techniques for specific domains.

(Xiao et al., 2023) addressed the issue of NLU models' energy consumption by introducing a bi-directional spiking neural network which transforms numeric values into discrete spiking signals and replaces massive multiplications with much cheaper additive operations.

2.2.3. Text To Speech

The final module necessary to build a VA is Speech Synthesis, aka Text To Speech (TTS), which gives vocal feedback to the user interacting with the system. Literature offers two possible types of solutions for this task: (i) a two-stage pipeline, where a spectrogram is generated (mel or Hz scale) first, then a voice-encoder (vocoder) produces audio by taking the spectrogram as input; (ii) an end-to-end approach that uses a single model to generate audio straight from the text. The first approach is the most widespread, and several models of this kind are available in the literature. Tacotron2 (Shen et al., 2018) is a neural network architecture provided by Google that maps characters to mel-scale spectrograms through a recurrent sequence-to-sequence feature prediction network. This is the most famous spectrogram generator, and its implementation has been picked up by many other contributors (e.g. (Nekvinda and Dušek, 2020; Valle et al., 2024),). The Google version achieves a Mean Opinion Score (MOS) of 4.53, which corresponds to professional speech. Regarding vocoders, they are divided into glow-based and Generative Adversarial Network (GAN) architectures. The former learn the data distribution and therefore minimizes the negative log-likelihood, while the latter learn to distinguish real data from fake samples produced by the generator, i.e., a minimax game on the classification error loss. Among glow-based vocoders we recall WaveGlow (Prenger et al., 2019), SqueezeWave (Zhai et al., 2020), and UniGlow (NVIDIA, 2023a); while GAN models are MelGAN (Kumar et al., 2019), and HiFiGAN (Kong et al., 2020). On the other hand, examples of end-to-end solutions include FastPitchHifiGAN (NVIDIA, 2023b), which combines a spectrogram generator, FastPitch (Łańcucki, 2021), and the HiFiGAN vocoder to generate a waveform from the text; FastSpeech2HiFiGAN

(NVIDIA, 2023c), which combines the FastSpeech2 spectrogram generator (Ren et al., 2021) and HiFiGAN into one model to be trained end-to-end. All of the above-cited models are available in the NeMo toolkit, which also provides some English pre-trained solutions. NaturalSpeech (Tan et al., 2024) is an end-to-end TTS system exploiting a variational autoencoder to directly generate the waveform. It achieves human-level performance on the LJ Speech dataset (Ito and Johnson, 2017). FastDiff-TTS (Huang et al., 2022) is another end-to-end TTS solution which provides for high-quality speech synthesis. Experimental results show also the inference speed of the model, which allows for real-time speech synthesis. Also in this case, the official source code has not been released yet.

A recent research specialization concerns accented TTS, which has significant real-world applications, particularly for endangered languages and dialects. Moreover, changing the accent intensity potentially allows specific users to understand its produced speech better (Liu et al., 2024) and accent modeling contributes to improved speech quality and better user experience (Zhou et al., 2024).

3. System architecture

This chapter provides an overview of the system architecture, to support a high-level understanding of components, interfaces, and overall design. Implementation details of the constituting modules are presented in the next chapter.

End-to-end VAs require comprehensive processing, from the reception of the incoming speech to the provision of feedback on the execution of the requested intent (i.e., action). Therefore, we designed a system architecture able to perform the following steps:

- Human voice detection;
- Activation keyword detection;
- Speech recognition and conversion into text;
- Text conversion into meaningful data;
- Text conversion into speech to give feedback to the user.

The order of the steps in a conversation with a VA is the following. First, the user's speech is detected and, if the uttered word is recognized as a wake-word, speech recognition is activated. Then, key information, such as intents and entities, is extrapolated from the inferred sentence to interpret the user's request. The detected request is finally executed by the system and feedback is provided in the form of a voice response from the assistant.

The conversational process can thus be implemented as the cascade of specialized modules, each one handling different parts of the conversation.

A VAD module is used to detect the human voice, distinguishing it from non-human background noise. The module is a binary classifier (classes: "voice" and "noise"). It must always be listening and needs deactivating only when the keyword is spotted, to be reactivated at the end of the conversation.

A KWS block activates the conversation with the VA only if a certain wake-word is uttered, otherwise, no action is taken. This module is a binary classifier as well, but it distinguishes the wake-word from all the other human-uttered words.

If the user pronounces the wake-word, an ASR module is activated, which transcribes the human-spoken sentence into text. This module is more complex than the previous two since it performs a multi-class classification on the input speech, where the classes are the characters of the target alphabet and the punctuation marks.

The obtained text string is then processed by an NLU block, which extrapolates intents and related entities. An intent is a group of utterances with similar meanings while entities are data containing significant values, that the system must consider like parameters for the execution requested by the user, such as numerical values or geographical locations. This step is necessary to obtain a semantic interpretation understandable by the VA system so that it can perform what the user requested.

After the execution of the intent, a TTS block is responsible for vocally notifying the user about the success or failure of the given command. This step, therefore, involves the use of a spectrogram generator, which converts the raw text of the system response into a representation of the spectrum of frequencies of the audio signal over time, followed by a vocoder, which synthetizes this representation into a speech waveform.

The above-presented task is complex, for an embedded environment, and the modules to achieve a level of performance similar to state-of-the-art of cloud-based applications are resource-demanding. Thus, we estimated minimum requirements such as 4 GB of memory, GPU not strictly required but suggested to achieve faster inference (at least 1 tera operations per second (TOPS)), and power consumption not higher than 30 W. Such requirements are met by last generation high-end embedded devices. As our development board, we chose the NVIDIA Jetson AGX Xavier. We chose this system because of its small size (105 mm $\times$ 105 mm $\times$ 65 mm) and relatively high computing power and GPU availability (Franklin, 2018). The board's specifications are reported in Table 2, which also shows the specifications of Amazon Alexa's cloud infrastructure (Amazon EC2 Inf1 (Amazon Web Services, 2019)), to highlight the gap between cloud-based and edge solutions in terms of computing capabilities.

The system architecture we propose, shown in Fig. 1, is aimed at using the VA in an environment where the Internet connection cannot be relied upon and low latency is required. Moreover, to keep a high level of extensibility, we designed it foreseeing a subsequent addition of further languages, as will be discussed in Section 4.5. The above-described conversation steps have been slightly streamlined, to make the system more suited for embedded deployment. Particularly, due to the similarity between their tasks (i.e., binary classification), we decided to implement the VAD and KWS tasks in a single module, namely Speech Classification (SC). This block performs a binary classification in which the two considered classes are the wake-word and its negation, which includes both non-human background noises and all other human-uttered words different from the wake-word. This choice allowed us to perform a faster inference while respecting privacy since the ASR block is not activated until the wake-word is recognized, which prevents the system from getting unintended information from the user.

4. In-car embedded VA implementation

After presenting the system architecture, now we delve into a comprehensive exploration of the proposed in-car embedded VA implementation.

To prevent overfitting and target generalization, an early stopping approach was employed for the training of each model. Early stopping (Prechelt, 1998) is a regularization technique that monitors a validation metric, such as validation loss or accuracy, and halts the training process when the metric fails to improve over a certain number of iterations. By utilizing early stopping, each model's training time was significantly reduced. All values of training epochs reported in this article have been obtained using early stopping.

Table 2
Comparison between target embedded device and Amazon Alexa's cloud specifications.

Feature	NVIDIA Jetson AGX Xavier	Amazon EC2 Inf1
Memory	32 GB	Up to 192 GB
GPU	512-core NVIDIA Volta GPU with 64 Tensor Cores (32 TOPS)	Up to 16 AWS Inferentia chips (128 TOPS each)
CPU	8-core NVIDIA Carmel ARM v8.2 64-bit	Up to 96 2nd gen Intel Xeon (x86 64-bit)
Power	Between 10 W and 30 W	Not specified

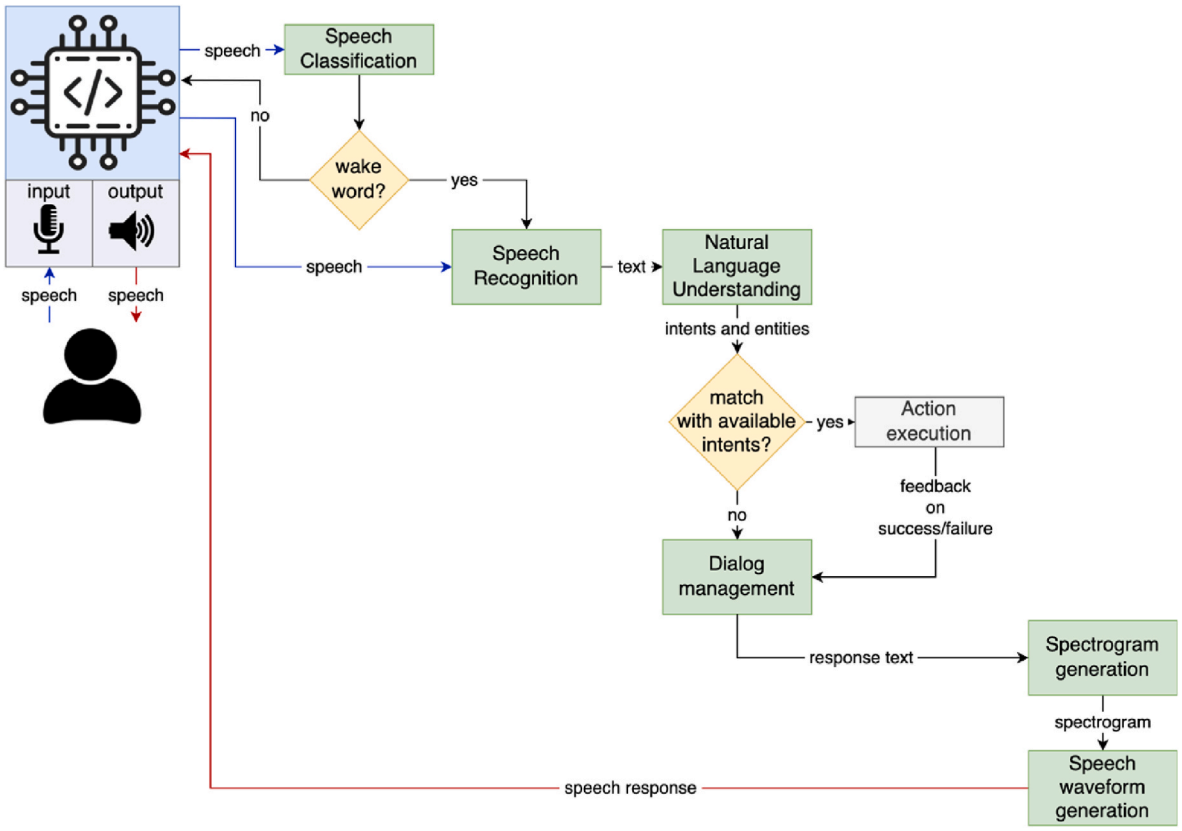

Fig. 1. System architecture of the VA.

4.1. Speech Classification (SC)

As shown in Fig. 1, the SC module is the first one in the pipeline and remains continuously active to catch the wake-word from the user. This module is only disabled during the actual dialogue between the user and the VA. For the implementation of this specific task, we opted for the MarbleNet model (Jia et al., 2021), a deep neural network composed of blocks of 1D time-channel separable convolution, batch normalization, ReLU, dropout layers, and Cross Entropy loss. This model, derived from the QuartzNet model (Section 4.2.1), has been developed by NVIDIA and is available inside the NeMo toolkit. We opted for this solution as it can achieve performance levels similar to state-of-the-art VAD models with about one-tenth of the parameters (88K versus the 738K of the CNN-TD model proposed in (Hebbar et al., 2019)). Its architecture, like all the VAD architectures, is built to perform a binary classification task: distinguish what parts of an audio segment contain human speech and what do not. To simplify our pipeline and speed-up inference, we decided to use this architecture differently. We changed the two class labels to distinguish the actual wake-word from other human-pronounced words and the background noise. For the wake-word class, we used a dataset provided by Lifetouch, an Italian hi-tech company in the automotive and transportation sector, while for the counterpart we created a dataset exploiting human-uttered words from the Lifetouch dataset and merged it with typical in-car background noises, downloaded from Freesound (Font et al., 2013) using its Application Programming Interface (API). These sounds include traffic noises, car, and bus noises, chatter, etc. In this way, we managed to perform VAD and KWS by using a single DL model.

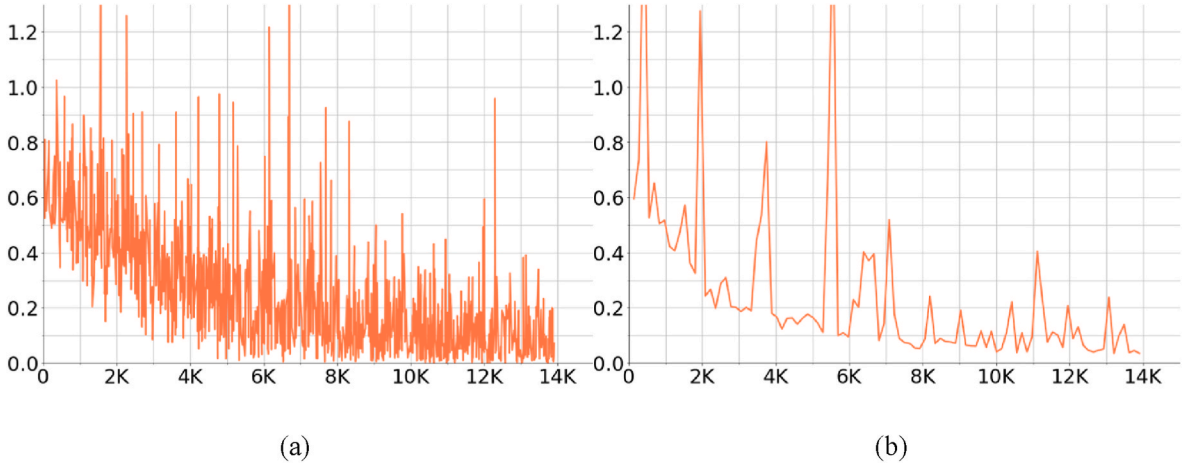

Fig. 2. MarbleNet model Cross Entropy (a) training and (b) validation losses (y-axes) over 97 training epochs (~14000 steps, x-axes).

We trained the model for 97 epochs (around 14,000 steps), achieving a 93% accuracy. Given the modest size of our dataset (i.e., around 400 wake-word training samples and 800 background noise samples), we argue that 97 training epochs are sufficient to achieve good accuracy while avoiding overfitting. This appears also in Fig. 2, where training and validation Cross Entropy losses are plotted over the training steps. The small gap between training and validation loss demonstrates a good model fitting.

4.2. Automatic speech recognition

ASR is a crucial step in the process, together with TTS, since these layers implement the main interface with the user. They require high-dimension datasets, with huge quantities of hours of speech, to achieve state-of-the-art performance. As mentioned, this is an issue for languages different from English or Chinese. For ASR, multiple-speaker datasets are also required. To alleviate the difficulty, the NeMo toolkit offers the possibility of TL, which leverages the knowledge embodied in an English pre-trained model, allowing the use of a smaller Italian dataset for fine-tuning the model's weights. Specifically, we started from an NVIDIA QuartzNet 15x5 model (Kriman et al., 2020) (Fig. 3), pre-trained in English, and performed fine-tuning on it using the Common Voice Italian dataset. We opted for the QuartzNet 15x5 model because of the built-in support for TL of the NeMo toolkit and the limited number of parameters (19M) compared to the state-of-the-art (Kriman et al., 2020).

4.2.1. QuartzNet 15x5 model

The QuartzNet model consists of blocks with residual connections in between. Each block is composed of modules with 1D time-channel separable convolutional layers, batch normalization, and ReLU layers. The network features an encoder and a decoder. The encoder handles acoustics and produces a hidden representation of the recorded voice. It can be viewed as an acoustic model for extracting speech features, which are then directly piped to a decoder that outputs text. The decoder takes this representation and generates letters according to the target language's alphabet. The encoder can therefore be reused across languages, while the decoder depends on the actual target alphabet.

The QuartzNet 15x5 model, trained with Connectionist Temporal Classification (CTC) Loss (Graves et al., 2006) on a combination of datasets (namely: on LibriSpeech (Panayotov et al., 2015), Mozilla Common Voice (Ardila et al., 2020), WSJ (Paul and Baker, 1992), Fisher (Cieri et al., 2004), Switchboard (Godfrey et al., 1992) and NSC Singapore English (Koh et al., 2019)), achieves a WER of 3.79% on LibriSpeech dev-clean, and a WER of 10.05% on dev-other sets, while having only 18.9M parameters (NVIDIA, 2023d). This model is part of the NVIDIA NGC collection (Tekur and Gardiner, 2019), available in the NeMo toolkit.

4.2.2. Mozilla Common Voice

For training our system, we relied on the Mozilla Common Voice dataset (Ardila et al., 2020), in its Italian version (Common Voice Corpus 8.0), which consists of a total of 310 validated hours (MP3 format). The size of the dataset and the availability of manifests useful for the training and testing phases, combined with the lower level of the other open-source datasets available, made us lean towards this choice for our TL task.

4.2.3. Transfer learning experiment

Once established the English pre-trained model and the Italian dataset, we started the TL of the model. For this purpose, we employed the NeMo toolkit, exploiting the QuartzNet 15x5 model.

A pre-processing of the audio files was necessary since the English model was trained with WAV files with a 16 kHz sampling rate, so the Common Voice clips (MP3 format, 48 kHz) have been converted to match the QuartzNet 15x5 training data. During this phase, JSON manifests for training, validation, and test phases have been created, specifying the name of the clip, the duration, and the sentence uttered. Other features, originally present in the TSV manifests but not useful for our purpose, were removed. Additionally, all the characters were converted to lowercase.

For the fine-tuning of the model, as suggested in (Huang et al., 2020) for Spanish and Russian, the NovoGrad optimizer was used, with β_1 and β_2 set to 0.95 and 0.25, respectively (Ginsburg et al., 2020). Also, the learning rate was set to 0.001, with a Cosine Annealing policy and a 12% warm-up ratio (Loshchilov and Hutter, 2017). The decoder's labels were changed to match the Italian alphabet. The network was trained with the help of the PyTorch Lightning tool (Falcon, 2015) for 256 epochs, with CTC Loss (as the original QuartzNet 15x5), batch size 32, and Automatic Mixed Precision (AMP) O1 (NVIDIA, 2018).

After 256 epochs and around 921,200 training steps, the results proved to be acceptable. Fig. 4 shows the WER of the model on the training and the validation datasets, respectively. The WER, which is the industry standard for measuring ASR models' accuracy (Urban and Mehrotra, 2023), is computed as follows:

$$WER = \frac{(i_w + s_w + d_w)}{n_w} \tag{1}$$

where n_w is the number of words in the reference text, s_w is the number of words substituted (in the inferred text), d_w is the number of words deleted, and i_w is the number of words to be inserted to transform the hypothesis text into the ground truth. Consequently, WER is a number between 0 and 1. The final validation WER is 11.7%. For reference, in speech transcription, WERs of 5% are considered professional level (Xiong et al., 2017), while 20–25% is the upper bound of acceptable performance (Munteanu et al., 2006).

In the next section, we will compare these WER results with other open-source offline solutions, showing that they are in line with, if not better than, state-of-the-art cloud/desktop computing implementations.

Fig. 5 shows the training and validation loss curves over the 256 training epochs. The training took 5 days and 21 h on two 24 GB NVIDIA GeForce RTX 3090 GPUs. The model effectively learned the underlying patterns and relationships in the data, as it appears from the training and validation loss plots. Training was stopped at 256 epochs, as the

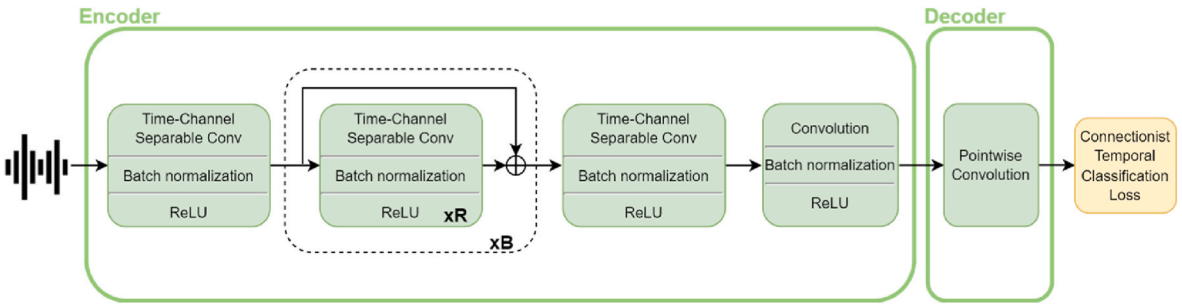

Fig. 3. QuartzNet B × R architecture.

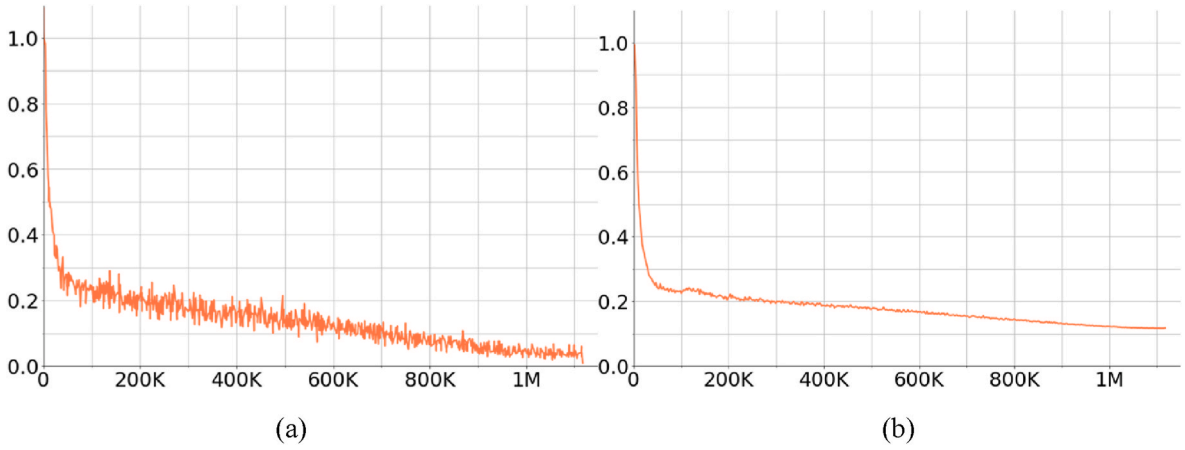

Fig. 4. The training (a) and validation (b) WERs (y-axes) over 256 training epochs (~921200 steps, x-axes).

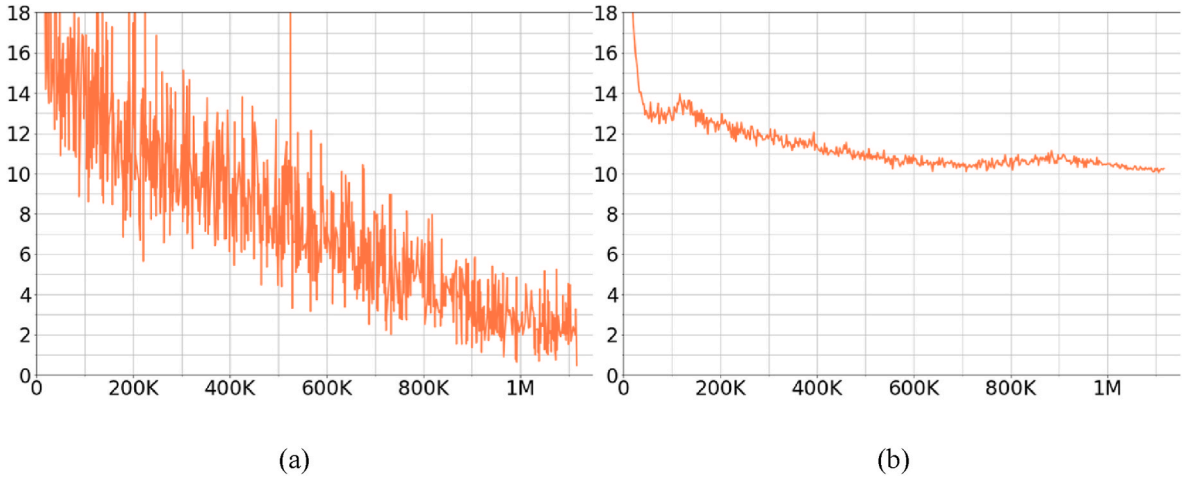

Fig. 5. QuartzNet model CTC (a) training and (b) CTC validation losses (y-axes), over 256 training epochs (~921200 steps, x-axes).

validation loss seemed not to improve further.

4.2.4. Comparison with other ASR models

We compared our trained model with the state of the art in terms of WER, Character Error Rate (CER), and transcription time. In particular, the comparison was performed with the Vosk and DeepSpeech models, which are both open-source and do not require any cloud service connection. The DeepSpeech model here considered has been obtained via TL, starting from the English DeepSpeech model, pre-trained with the Common Voice English dataset, and then trained with the Common Voice Italian dataset, like our model.

To make the test as fair as possible, we used the Common Voice test set, which was never used before by any of the models. This dataset is composed of 12,928 utterances, resulting in about 12 h of speech in total. We performed the calculation of WER and CER using the Python JiWER tool (Vaessen, 2022). Results are shown in Table 3.

The length of the audio file considered is 5.269s.

The table shows that our model achieves WER and CER values that are acceptable according to (Munteanu et al., 2006) (upper acceptability limit: 20–25%), and lower than both the Vosk and the DeepSpeech models. On the other hand, the QuartzNet 15x5 English model (trained with more than 3300 h of English spoken language) achieves a WER of 3.79% on LibriSpeech dev-clean, but a WER of 10.05% on dev-other sets, which also hints at a significant variability of the performance depending on the test set.

Taking into account the variety of word lengths in the Italian vocabulary, we also evaluated our model with CER, which considers the error at a character level instead of at a word level. According to the WER, a word is considered incorrectly recognized if just one letter in it is not correct. For the CER, it is obtained using the following equation:

$$CER = \frac{(i + s + d)}{n} \tag{2}$$

where n is the total number of characters and i, s and d refer to the number of character insertions, substitutions, and deletions required to transform the hypothesis text into the ground truth one, respectively. The CER obtained by our model confirms the considerations made on the WER, showing however how the DeepSpeech model, considering this type of error, comes closer to the other two. This behavior is in line with the one reported in (Lund et al., 2013), where the nonlinear relation between WER and CER is shown in Fig. 1.

Table 3
Word and Character Error Rates and transcription times.

Model	Dataset	WER (%)	CER (%)	Transcription time [a] (s)
Ours (based on NeMo)	Common Voice Italian test set	11.7	3.12	0.215
Vosk	Common Voice Italian test set	29.8	12.5	0.464
DeepSpeech	Common Voice Italian test set	45.8	13.24	1.778

While dealing with the accuracy of the model, another aspect to be taken into consideration is the type of test set being used. Taking a look at the utterances of which it is composed, we noticed that there are many obsolete words in today's Italian language, together with some non-Italian words, or proper names. Such sentences as, e.g., "Tom Sawyer e l'amico Huckleberry Finn sono testimoni di un omicidio" ("Tom Sawyer and his friend Huckleberry Finn are witnesses of a homicide"), are difficult to identify and transcribe correctly for each of the Italian models and can contribute to raising the level of both WER and CER. This also occurs in the training set, where such types of utterances, present minimally, do not contribute effectively to training the model for use with a common language. If the utterances of uncommon use were not included in the dataset, the performance of all three models, in terms of WER and CER, would have probably increased. We argue that this modification to the dataset would make particular sense for tasks such as in-car VAs, which will hardly involve the user entering uncommon words.

For the sake of comparison, Fig. 6 reports WER benchmarks provided by Picovoice (2022b) for the English idiom, together with results for Italian (already shown in Table 3). Both tests were performed on the Common Voice dataset (English and Italian versions respectively). The orange bars in the figure represent cloud ASR solutions (i.e., Amazon Transcribe, Azure STT, Google STT (two versions), IBM Watson STT), while the blue bars (i.e., the two Picovoice models, the two DeepSpeech models, the Vosk model, and our model) are not cloud-connected. Our model achieves a WER comparable to that obtained by cloud-based models in English. We highlight that our model has been fine-tuned from English with a relatively small Italian dataset, so the fact that its performance is comparable, if not better than English and cloud-connected models is significant.

The last aspect we considered for the assessment was the transcription time. This is particularly important for using the model within ASR systems that require acceptable accuracy along with low latency (e.g., for real-time tasks such as navigation). Table 3 shows the results for an audio file of about 5 s, for which our model outperforms the other two also in terms of transcription time. The DeepSpeech model takes about 8 times the time of our NeMo model, not proving itself to be suitable for real-time applications.

4.3. Natural language understanding

The NLU module handles the conversion of raw text into structured data, i.e., intents and entities, used by the system to classify the sentence based on its content. NLU differs from NLP since the former interprets the significance of the sentence, whereas the latter only converts the whole text into its semantic pieces. The tool we chose for NLU is Rasa (Bocklisch et al., 2017), due to its high flexibility and customizability, given the possibility of creating and exploiting specific NLU pipelines with the desired modules. The elements composing our NLU pipeline are shown in Fig. 7 and explained below.

- WhitespaceTokenizer: splits the sentence into words separated by whitespaces;
- RegexFeaturizer: creates features for entity extraction and intent classification by looking for regular expressions previously defined in the training set;
- LexicalSyntacticFeaturizer: moves over the sentence with a sliding window and extrapolates lexical and syntactic features;
- CountVectorsFeaturizer: transforms a given text into a vector based on the frequency of each word occurring in the text. Word token counts are used as features;
- Dual Intent Entity Transformer (DIET) Classifier: a transformer-based (Vaswani et al., 2017) architecture, which handles both intent classification and entity recognition (Bunk et al., 2020);
- Entity synonym mapper: if the training data contains words that can be defined as synonyms of other words, this component will make sure that the detected entity values will be mapped to the same value;
- FallbackClassifier: classifies a message with the intent "nlu_fallback" if confidence is under a given threshold (we set it to 0.3). A fallback will also occur if the two top-ranked intents have similar confidence.

As an alternative, we tried to start from a SpaCy pre-trained model and make use of the related pipeline but achieved worse performance. We argue that this is due to the nature of our use case. Given our need to develop an NLU engine with very specific terminology, i.e., in-car intents only (Table 1), a generic pre-trained model is not well-suited for this kind of scenario, as confirmed by the Rasa documentation as well (Rasa, 2024).

Based on the requirements described in Table 1, we defined 85 intents, and, for each one of them, an average of five example sentences to be used in the training set (written in YAML format). An example of setting a radio's volume is reported in Fig. 8. As it appears, significant values such as numerical quantities are reported following a specific syntax which allows the system to recognize them as entities and so to extrapolate their value to perform the requested action. For the case taken into consideration, the requested radio volume is marked as an entity so the system can set the new desired value.

We trained the DIET classifier for 83 epochs, achieving 98% accuracy. Training and validation loss graphs are reported in Fig. 9.

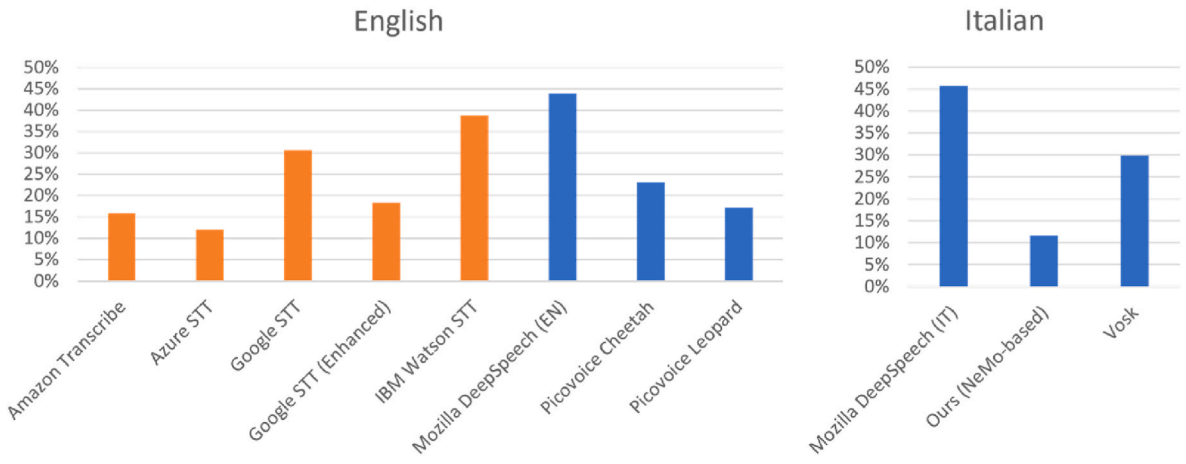

Fig. 6. WER comparison of some cloud-connected (orange bars) and edge (blue) solutions tested on the English and Italian portions of the Common Voice dataset (English data provided by Picovoice (Picovoice, 2022b)).

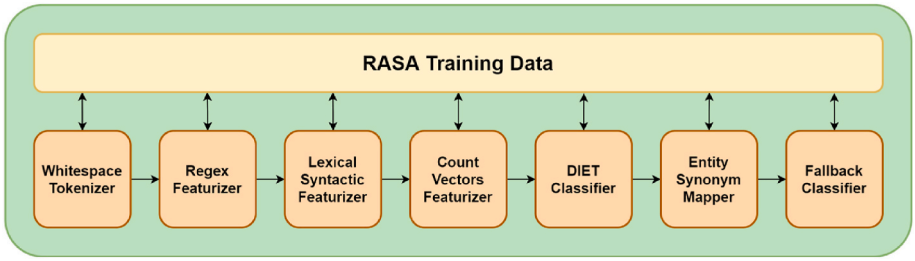

Fig. 7. The Rasa pipeline.

```
- intent: set_volume
  examples: |
    - imposta volume a [3](volume_level)
    - imposta volume radio a [4](volume_level)
    - imposta volume musica  a [5](volume_level)
    - imposta a [3](volume_level) volume
    - imposta a [4](volume_level) volume radio
    - imposta a [5](volume_level) volume musica
```

Fig. 8. Rasa intent declaration.

4.4. Text To Speech

The TTS module handles the generation of speech from an input text and is composed of two submodules: a spectrogram generator, which generates a mel (a perceptual scale of pitches judged by listeners to be equal in distance from one another) or a Hz spectrogram from the input text; and a vocoder, which generates the speech from the spectrogram. These two submodules can be trained separately or together, as previously described in Section 2. Due to the reliability of the Tacotron2 model (Valle et al., 2024), its open-source availability, its results in terms of MOS (Shen et al., 2018), and the integration inside the NeMo toolkit, we opted for this spectrogram generator for our purpose. As per the vocoder, we empirically tested all the models available inside NeMo: WaveGlow, SqueezeWave, UniGlow, MelGAN, and HiFiGAN. We opted for the one that gave the clearest output voice: MelGAN. The last component to be chosen was the dataset. As for the ASR case, the availability of open-source Italian datasets is limited, but M-AILABS (Solak, 2019) turned out to be reliable because of the availability of a sufficient number of hours of speech from a single speaker (18 h in total for the male speaker). The resulting TTS module's architecture is shown in Fig. 10.

Tacotron2 has a sequence-to-sequence architecture. It consists of an encoder, which creates a hidden representation of the input alphabet characters, and a decoder, which transduces this representation into a mel spectrogram. After projecting inputs and location features to 128-dimensional hidden representations, the encoder output is given to an attention network that summarizes the encoded sequence as a context vector. Location-sensitive attention is used (Chorowski et al., 2015), which defines the part of the encoder data to be used at each decoder step. The decoder is an autoregressive recurrent neural network that predicts the mel spectrogram from the encoded input sequence. Its output is an 80-dimensional audio spectrogram with frames computed every 12.5 ms, capturing the pronunciation of words, volume, speed, and intonation.

We trained the Tacotron2 model for 1500 epochs (around 73,500 steps). Results are reported in Fig. 11 in the form of a sample mel spectrogram's ground truth and prediction, while Fig. 12 reports the training and validation loss curves, that show a rapid convergence of the model.

The two generated spectrograms are quite similar, and this is also confirmed by the alignment plot, shown in Fig. 13. The encoder (y-axis) takes an input character and its current state at each step, to output a real vector representing the status of the network at that moment. The training audio sample length here is about 60, so there are about 60 vectors generated by the encoder. The decoder (x-axis) takes all the vectors (the y-axis) to generate audio frames (mel-spectrogram). The decoder also works step-by-step, at each step (around 200 steps) it decides which vectors (on the y-axis) are important to create audio frames at that particular moment. An almost diagonal line results when audio frames are created by focusing on the correct input characters in order.

As for the vocoder, MelGAN is a non-autoregressive feed-forward CNN to transform mel spectrograms into time-domain waveform samples in a GAN setup. However, unlike traditional GANs, MelGAN does not use a global noise vector as input due to a difference in the generated waveforms when additional noise is fed to the generator. The generator is a fully convolutional feed-forward network that up-samples the input

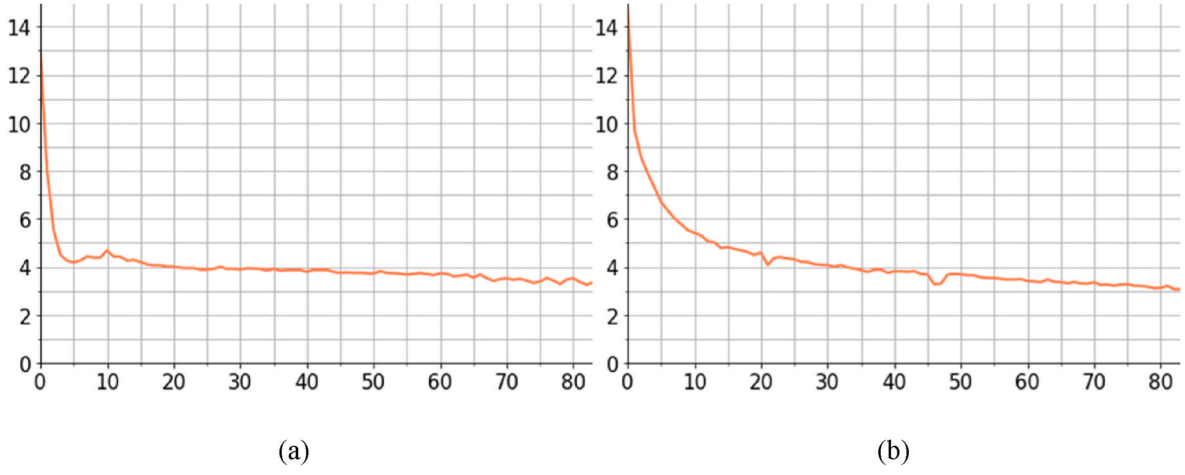

Fig. 9. DIET model dot-product training (a) and validation (b) losses (y-axes) over 83 training epochs (x-axes).

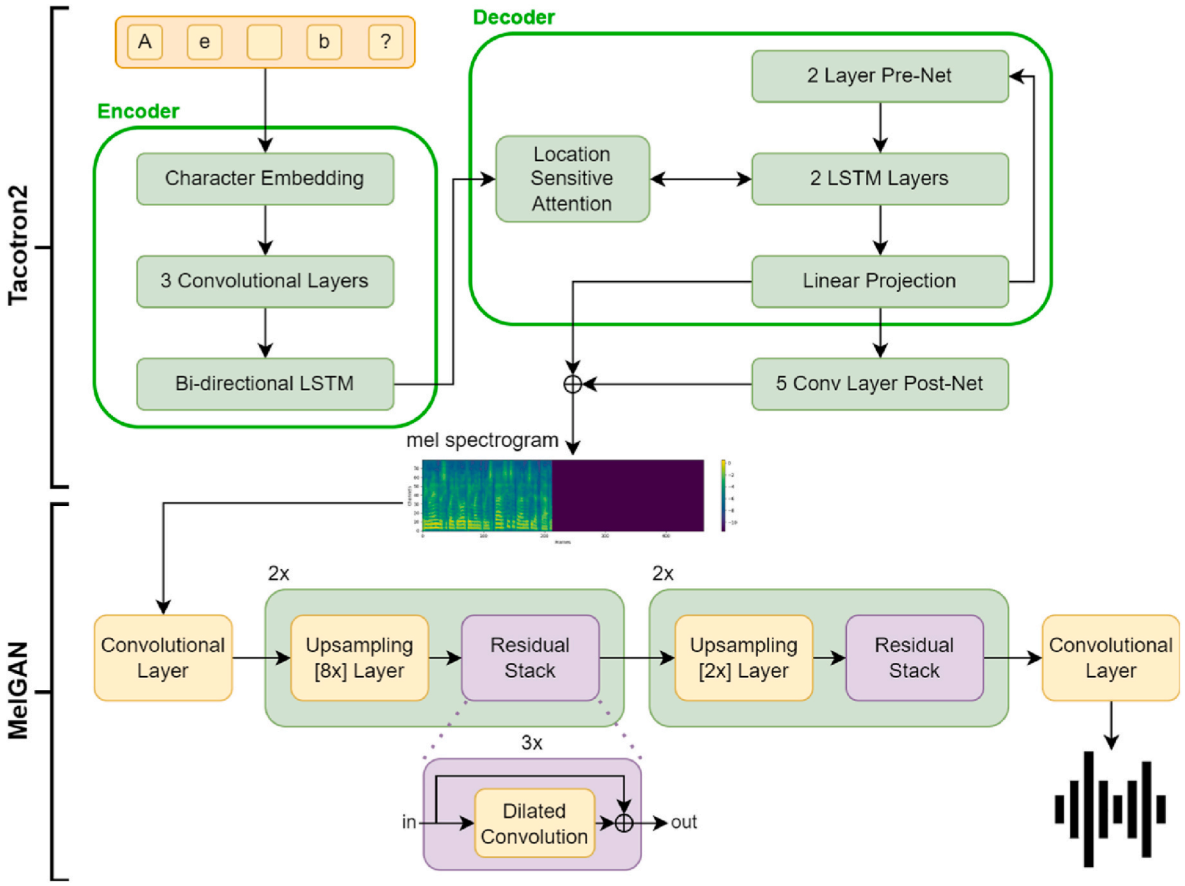

Fig. 10. The TTS module architecture, consisting of the Tacotron2 spectrogram generator and the MelGAN vocoder.

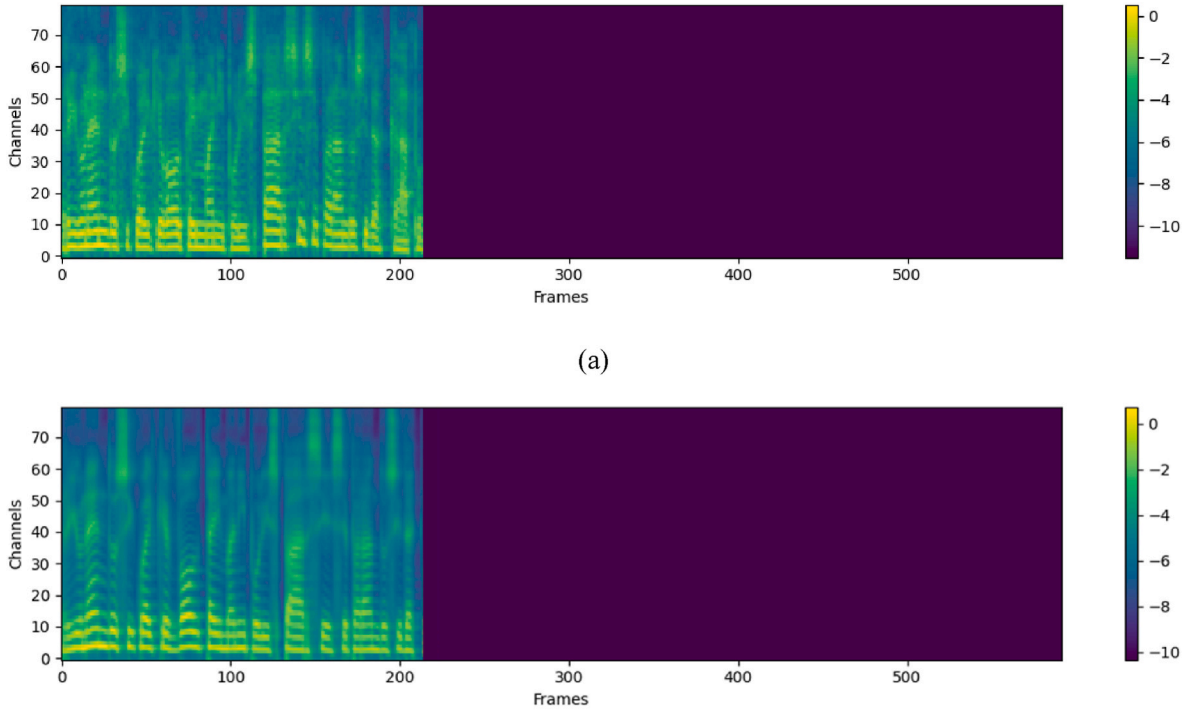

Fig. 11. The Tacotron2 target (a) and predicted (b) mel spectrograms after 1500 epochs (~73,500 steps).

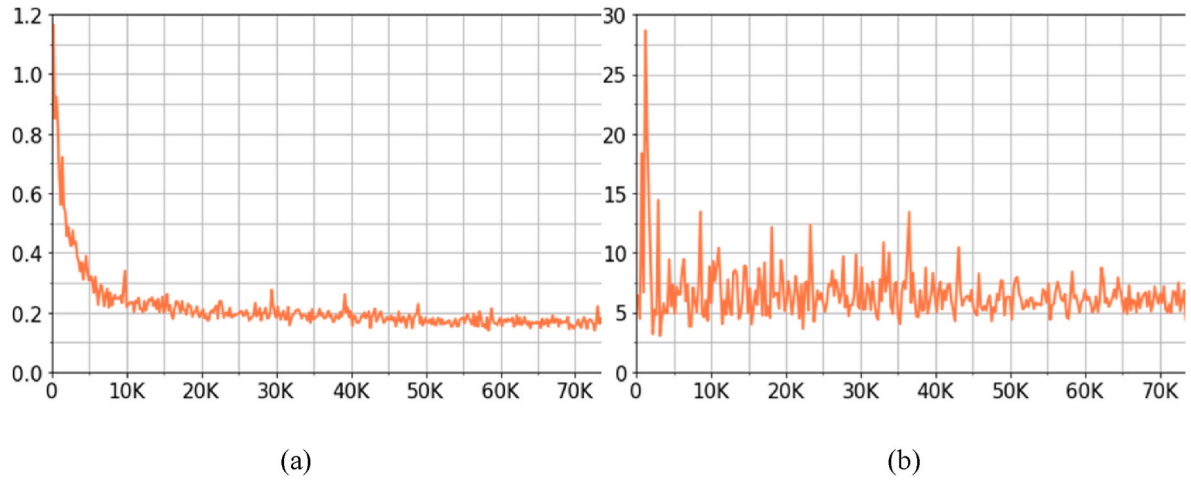

Fig. 12. Tacotron2 model mean squared error training (a) and validation (b) losses (y-axes) over 1500 training epochs (~73,500 steps, x-axes).

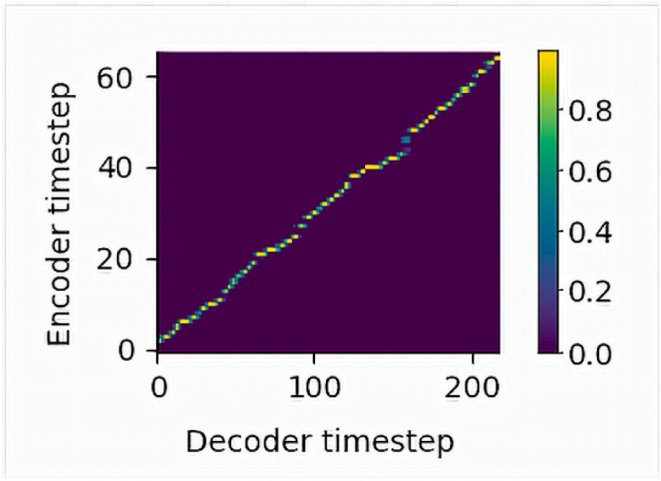

Fig. 13. Alignment plot of the Tacotron2 model after 1500 epochs (~73500 steps).

sequence with 256x multiplicative factor in 4 stages (8x, 8x, 2x, 2x, since the mel-spectrogram is at a 256x lower temporal resolution). The discriminator block has a multi-scale architecture composed of 3 discriminators that have the same structure but operate on different audio scales (raw audio, raw audio down-sampled by 2, and by 4). This way, each discriminator learns discriminative features based on a specific target frequency range (e.g., the discriminator operating on down-sampled audio does not have access to high frequencies, so it tunes for low frequencies only).

We trained the MelGAN model for 2950 epochs (approx. 108,000 training steps). In Fig. 14, a comparison between the target and predicted plots is reported, demonstrating the similarity between the two. Fig. 15 shows the model's training and validation losses. The generator's training loss (Fig. 15a) has a negative spike at about 10,000 steps. This is due to the introduction of the discriminator (Fig. 15b), which aims to discern between waveforms produced by the generator and ground-truth waveforms. The convergence of the overall model is apparent from Fig. 15c (validation): while the discriminator gradually enhances its ability to distinguish between real and generated waveforms, the waveform generator's loss remains stable.

4.5. Toolchain composition

After the definition of the single blocks composing the toolchain, we connected them as shown in Fig. 16.

To keep the SC module always active to catch the wake-word from the user and activate the subsequent modules only after the wake-word has been detected, we kept the two audio streams separated. This ensures that the ASR audio stream is activated if and only if the wake-word is uttered. The PyAudio library was used for this purpose (Pham, 2006). PyAudio provides Python bindings for the audio I/O library PortAudio, a very simple API for recording and/or playing sound using a simple callback function.

When the wake-word is detected by the system, the ASR audio stream is activated, and the user command is transcribed into text in real-time. Since the Rasa toolkit does not accept numerical values written in letters inside sentences (as it is the output of the ASR model), we also developed a small Python library to convert numbers written in letters into digits. Supported values are from 0 to 999 since we did not need to expand it further for the in-car use-case (i.e., numerical values are for instance the radio volume or the interior temperature, which never exceed one thousand). The text is therefore passed to the NLU engine, which converts it into meaningful data, i.e., intent and entities. In a real application scenario, an intent execution block is supposed to execute the command (not covered here). Consequent feedback is given to the user based on the recognized intent and the outcome of its execution. For this purpose, we created a JSON file containing answers for each possible intent. Rasa itself can manage responses, but we created a dedicated JSON file to keep a higher level of versatility, particularly for adding new languages to be supported by the system. Thanks to this introduction, the insertion of a new language is made easier, since the user is only required to write the intents' responses inside the JSON file and the intents' sample sentences in the Rasa project to be trained. Also, this allowed us to use only the NLU part of Rasa and leave aside the Core, which handles responses and stories (sequences of questions-answers), thus streamlining the model and speeding up the inference times.

Contrary to Rasa, our Tacotron2 model does not handle numbers written in digits, since the training set we used (i.e., M-AILABS) has them written in letters. Thus, we converted them into letters using num2words (Savoir-faire Linux, 2022, p. 2), an open-source Python library supporting 38 languages, including Italian. A response is then given to the user and, if the system fails to parse the command, or the intent is inexistent, a default response is given, asking the user for a rephrase. In this case, the VA will not return to its idle state (i.e., waiting for the wake-word), but will wait for another command from the user, with a 10-s timeout, after which it stops listening.

The models related to the various blocks composing the toolchain are interchangeable, it is also easy to substitute them with newly-trained ones or to add models trained in other languages without having to

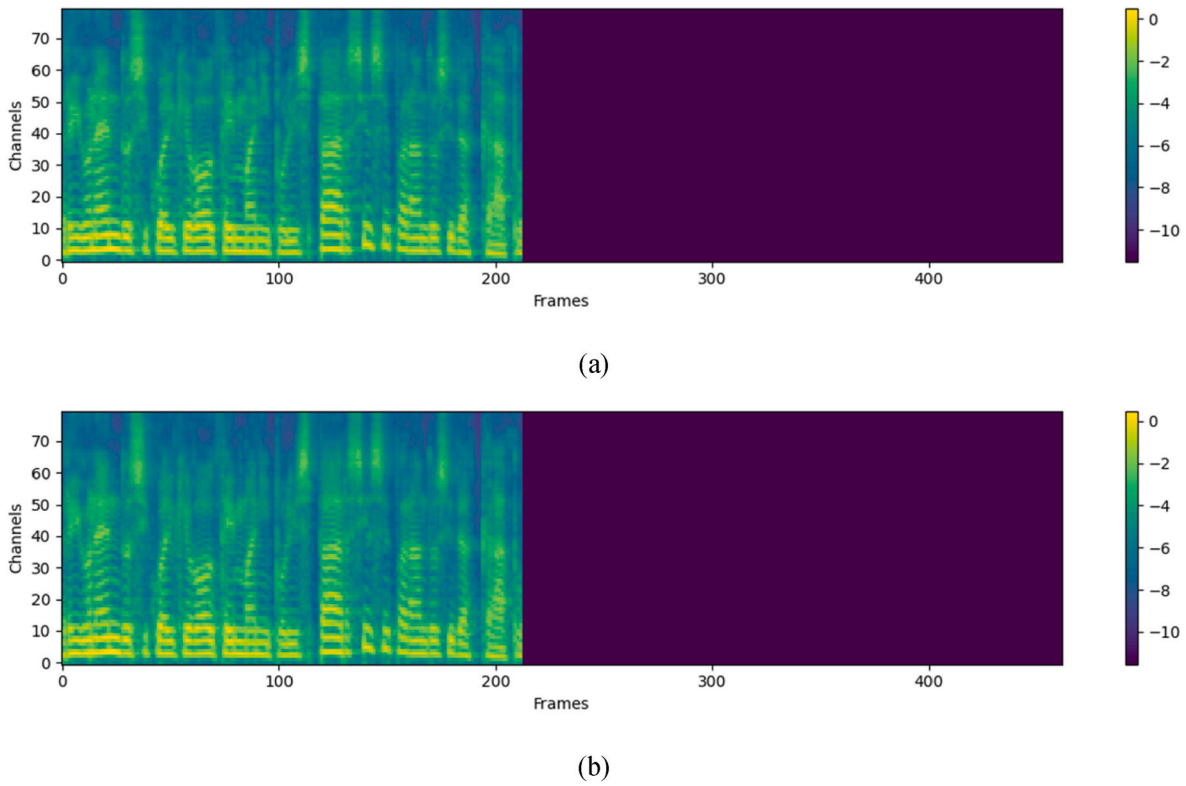

Fig. 14. The MelGAN target (a) and predicted (b) mel spectrograms after 2950 epochs (~108,000 steps).

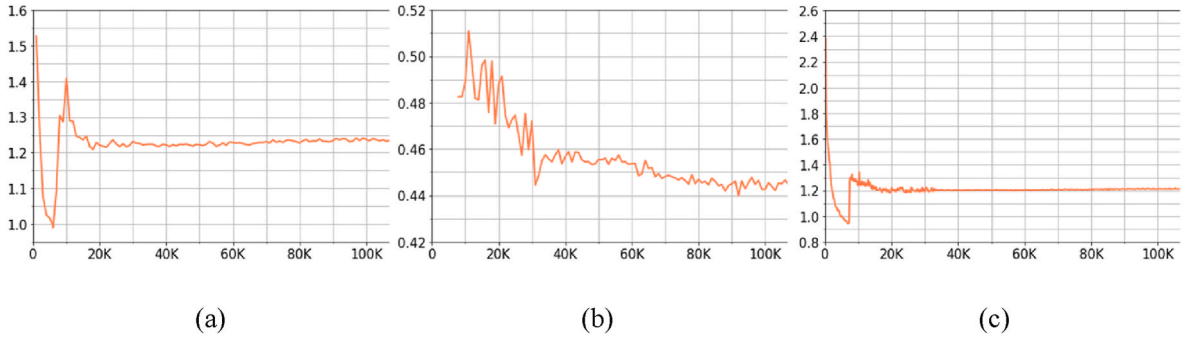

Fig. 15. MelGAN model training loss of the generator (a) and of the discriminator (b), and validation loss of the waveform generator (c) (y-axes) over 2950 training epochs (~73,500 steps, x-axes).

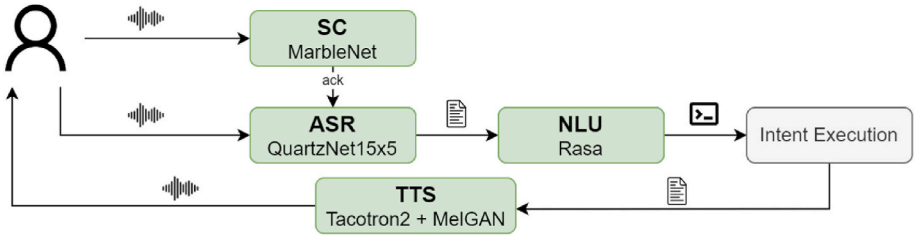

Fig. 16. High-level block diagram of the proposed toolchain.

put hands into the code. The user is only required to specify the new model's name in a JSON configuration file along with other important parameters (e.g., the expected sample rate or the model's language id, Fig. 17), assuring easier accessibility. Also, since the system is designed to support multiple languages, the language switch is handled directly in the configuration file, as well.

5. Results

This Section provides the performance analysis results of the system we developed to bring the advantages of offline computation in an in-vehicle VA. We deployed the VA system on an NVIDIA Jetson AGX Xavier board (Section 3). Since the board's architecture is a 64-bit ARM,

```
{
    "default_language": "IT",
    "led_pin": 7,
    "sample_rate": 16000,
    "vad": "vad100.nemo",
    "asr": "stt_quartznet15x5.nemo",
        "nlu": "nlu-20220321-105854.tar.gz",
    "spec": "tts_tacotron2.nemo",
    "vocoder": "tts_melgan.nemo",
    "ww_rec_audio": "ww_recognized.wav"
}
```

Fig. 17. Some parameters configurable from the JSON file.

and both Rasa and NeMo are not natively compatible with it, we built them from source.

As reported in Section 4, we identified and trained neural models that could achieve state-of-the-art performance in the reference test sets. In this section, on the other hand, we assess the system as a whole, starting with the inference times, that are reported in Table 4. Results show that the initialization time is quite long, but initialization is required only when the system is booted. Once the VA is running, the total running time is around 1 s (average computed over 20 runs, with different types of commands, of various durations).

Table 5 reports information about the VA memory occupation inside the Jetson AGX Xavier board. The table shows that the space on the disk required by the VA is approximately 4 GB, including all the required Python libraries and the DL models, which use only a few hundred MB (236.7 MB). Regarding the RAM usage, 6 GB is needed for the VA to perform inference, also including the initialization of the system and the two PyAudio streams (one for SC, one for ASR). Therefore, we argue that the system could work properly also on a lower profile board, at the price of higher inference times probably due to GPU constraints (i.e., lower TOPS than the Jetson AGX Xavier).

For the sake of overall system performance assessment in the target application domain, we tested the whole system on a custom dataset, consisting of 135 sentences recorded in a noiseless environment and 135 in a noisy one (i.e., in traffic), for a total of 270 utterances from 9 different speakers (3 females and 6 males). To cover all the types of use-cases presented in Table 1, we asked the speakers to record commands related to the application domain. Each user recorded 30 sentences, 15 noiseless and 15 in a noisy environment, semantically assimilable to one of the 85 intents present inside the Rasa training set (e.g., Fig. 8). This way, we managed to test all the designated VA's features even with a relatively small amount of audio clips. Some sample sentences extrapolated from the dataset are reported in Table 6 along with their ASR transcription, the ground truth intent, the NLU interpretation (i.e., inferred intent and, if present, entities), and the TTS-associated response. Particular attention was paid to critical cases. Errors are highlighted in italics. It can be seen that the first sentence reported ("Raise the temperature by eight degrees", in English)is parsed correctly. Nevertheless, a typical error is the recognition of an intent that is the opposite of another, i.e., "on" instead of "off", or "up" instead of "down". This appears from the second reported sentence("Turn off the left front seat heating", in English), which is similar to the first one (since both have an opposite version as well) but, in this case, "off" is confused with "on". We argue that this is due to the similarity between sentences, that differ only in the verb. This error can be avoided (or reduced) by adding domain knowledge to the system. However, the example is significant, as it shows the difficulty for the ASR module to recognize the [ɲ] phoneme (represented by the "gn" digram), which is present in the Italian language, but not in English. This hints at a possible weakness of the TL approach from English to Italian. In this case, we expect that the lower layers of the neural network should be updated, which makes the training on the target language more difficult. We thus argue that the availability of neural models deeply trained in some more languages than English and Chinese would improve the effectiveness of TL, by supporting a high variety of detectable phonemes. On the other hand, the third sentence reported shows that the system can extrapolate the correct intent even when a sentence is not completely correctly parsed.

Table 4
Breakdown of VA execution times.

Task	Description	Avg time (s)
Initialization	Start-up of models (pre-downloaded) and audio streaming. Occurs once, at booting time	57.691
Speech Classification	Detection of the wake-word from the user	0.023
ASR transcription time	Time elapsed between when the user stops speaking and when the ASR model obtains the entire sentence (ASR works through PyAudio streaming while the user speaks)	0.132
Intent Recognition	Convert written text to meaningful data	0.071
TTS	Convert raw text to mel spectrogram, then to speech	0.790

Table 5
Memory occupation of the VA.

Model	Disk occupation	RAM occupation at runtime
SC	361.5 KB	0.4 GB
ASR	72.5 MB	1.2 GB
NLU	25.4 MB	1.4 GB
TTS	105.5 MB (Tacotron2) + 32.9 MB (MelGAN)	1.4 GB
Net total	236.7 MB	4.4 GB
Gross total	~4 GB (including required libraries and models)	6 GB (0.6 GB init + 1 GB PyAudio streams)

For the evaluation, in addition to WER and CER (that concern the performance of the ASR model only), we have considered three additional metrics (Saxon et al., 2021), which take into account the NLU model as well, thus considering the execution of the whole system:

- Intent Classification Error Rate (ICER), the ratio of incorrect intent predictions to the total number of utterances;
- Slot Error Rate (SER), the ratio of incorrect entity predictions to their total number;
- Interpretation Error Rate (IRER), the ratio of the number of incorrect interpretations (i.e., sentences where either an entity or the intent prediction is incorrect) to the total number of utterances. This "exact match" is the strictest metric and the most relevant at the application and system level (Saxon et al., 2021).

Results are presented in Table 7.

Table 7 shows that the WER and CER values are higher than those reported in Table 3. We argue that this is essentially due to the quality of the audio files under test. Table 3 refers to audio files taken from the Common Voice dataset, which were selected after a crowd-sourced quality assessment phase, Table 7, instead, refers to audio clips that were recorded in an ecological context. For as IRER is concerned, our 9.77% result meets the user requirements specified for the project. This value becomes even lower if we do not consider the critical sentences described before (i.e., those that can also have an opposite meaning depending on a single word, usually the verb): 3.37% in the noiseless condition and 6.74% in the noisy one. IRER is significantly lower than the 22% error reported by (Perera et al., 2018) on Alexa, arguably on a more complex speech processing task (and in a different language, English instead of Italian). We are not aware of other comparable system-level results published in the literature. Values obtained in the noisy condition are not much worse than the others, which argues for

Table 6
Sample sentences for the VA's performance evaluation.

Spoken command	ASR transcription	Ground truth intent	Inferred intent	Entities	TTS response
Alza la temperatura di otto gradi (Raise the temperature by eight degrees)	Alza la temperatura di otto gradi	turn_up_temperature	turn_up_temperature	temperature = 8	Alzo la temperatura dell'aria condizionata di otto gradi (I turn up the air conditioning temperature by eight degrees)
Spegni il riscaldamento sedile anteriore sinistro (Turn off the left front seat heating)	*Speni* il riscaldamento sedile *e* anteriore sinistro	turn_off_seat_heating	*turn_on_seat_heating*	seat_type = anteriore sinistro	Accendo il riscaldamento del sedile anteriore sinistro (I turn on the left front seat heater)
Quanti litri consumo ogni cento chilometri percorsi? (How many liters do you consume per hundred kilometers traveled?)	Quanti *detre* consumo ogni 100 chilometri per *corsi*?	show_fuel_consumption_100 km	show_fuel_consumption_100 km	None	Il consumo è di quattro litri ogni 100 chilometri (Consumption is 4 L per 100 km)

Table 7
Error rates in noiseless and noisy environments.

Error type	Noiseless env	Noisy env
WER	17.71%	21.30%
CER	3.7%	5.1%
ICER	9.02%	9.35%
SER	3.09%	7.04%
IRER	**9.77%**	**13.08%**

the robustness of the proposed system to background noise, which is a key feature, especially in a vehicular context.

As a final summary, Table 8 presents a high-level comparison between our model and the state-of-the-art solutions, which shows that our model offers entirely offline end-to-end speech processing without loss of functionality, advancing the current state-of-the-art.

6. Conclusions and future work

This paper has investigated feasibility of an embedded end-to-end VA, by porting state-of-the-art AI methods in a relatively low-resource embedded environment. The system can work entirely offline, utilizing DL models properly optimized for embedded devices. The proposed system supports Italian language and reaches performance levels that make it a solid alternative to cloud-connected solutions while offering advantages such as reduced latency, lower energy consumption, and fewer privacy issues. This achievement has been made possible also through the application of the TL paradigm which, applied to the ASR model, allowed leveraging the knowledge embodied in an English pre-trained model, resulting in WER and CER values comparable to cloud-based solutions (Fig. 6).

Interactions between the system modules have been carefully managed with a dedicated system architecture (Fig. 1), which can be easily extended to support other languages through a configurable JSON file.

To the best of our knowledge, this work presents the first literature contribution of an end-to-end VA system architecture that supports a non-mainstream language like Italian, and results in the proposed vehicular application are promising. A target for future research is to reduce the system's latency, which currently stands at around 1 s. We also argue that an important advancement to decrease the possible input error to the NLU module could be the insertion of a text spelling correction module between the ASR and the NLU module. Additionally, future work will involve testing the presented toolchain on languages other than Italian and exploring domains beyond the vehicular context, as well as evaluating the system on lower-profile embedded boards. Finally, Large Language Models (LLM) are being employed also for VAs (Mahmood et al., 2023; Shafeeg et al., 2023; Vu et al., 2024), and their implementation on resource-constrained devices should be carefully considered, particularly to increase performance and versatility in intent recognition and generation of user responses.

Table 8
Functional feature comparison of VA systems.

Model	Multi-language	Offline operation	SC	ASR	NLU	TTS
Google Assistant	☑	limited	☑	☑	☑	☑
Siri	☑	limited	☑	☑	☑	☑
Picovoice	☑	☑	☑	☑	☑	×
Vosk	☑	☑	×	☑	×	×
DeepSpeech	☑	☑	×	☑	×	×
Ours	☑	☑	☑	☑	☑	☑

CRediT authorship contribution statement

Luca Lazzaroni: Investigation, Software, Writing – original draft. **Francesco Bellotti:** Conceptualization, Methodology, Writing – review & editing. **Riccardo Berta:** Conceptualization, Methodology, Supervision, Validation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The authors do not have permission to share data.

Acknowledgments

The authors would like to thank the LifeTouch team for the support and helpful feedback as well as for the possibility given to us to test our model in a real application context of noticeable interest for research such as offline VAs in vehicles. This project has received funding from the Piemonte Region's PRISM-E POR FESR 2014/2020 programme under grant agreement ADINA, code 337–151. The sole responsibility of this publication lies with the authors.

References

Alpha Cephei, 2020. VOSK offline speech recognition API [WWW document]. VOSK offline speech recognit. API. URL. https://alphacephei.com/vosk/, 12.14.21.

Amazon Web Services, 2019. Amazon EC2 Inf1 instances [WWW Document]. Amaz. Web Serv. Inc. URL https://aws.amazon.com/ec2/instance-types/inf1/. . (Accessed 4 November 2022).

Andrade, M., Wanderley, E., Azevedo, M., Medeiros, T., Silva, M., Silva, I., Costa, D.G., 2023. A voice-assisted approach for vehicular data querying from automotive IoT-based databases. In: 2023 Symposium on Internet of Things (SIoT). Presented at the 2023 Symposium on Internet of Things (SIoT), pp. 1–5. https://doi.org/10.1109/SIoT60039.2023.10389856.

Ardila, R., Branson, M., Davis, K., Henretty, M., Kohler, M., Meyer, J., Morais, R., Saunders, L., Tyers, F.M., Weber, G., 2020. Common voice: a massively-multilingual speech corpus. ArXiv191206670 Cs. https://doi.org/10.48550/arXiv.1912.06670.

Baevski, A., Zhou, H., Mohamed, A., Auli, M., 2020. wav2vec 2.0: a framework for self-supervised learning of speech representations. https://doi.org/10.48550/arXiv.2006.11477.

Berta, R., Bellotti, F., De Gloria, A., Lazzaroni, L., 2022. Assessing versatility of a generic end-to-end platform for IoT ecosystem applications. Sensors 22, 713. https://doi.org/10.3390/s22030713.

Biswas, M., 2018. Microsoft Bot framework. In: Biswas, M. (Ed.), Beginning AI Bot Frameworks: Getting Started with Bot Development. Apress, Berkeley, CA, pp. 25–66. https://doi.org/10.1007/978-1-4842-3754-0_2.

Bocklisch, T., Faulkner, J., Pawlowski, N., Nichol, A., 2017. Rasa: Open Source Language Understanding and Dialogue Management. https://doi.org/10.48550/arXiv.1712.05181.

Brinckhaus, E., Barnech, G.T., Etcheverry, M., Andrade, F., 2021. RoboCup@Home: evaluation of voice recognition systems for domestic service robots and introducing Latino Dataset. In: 2021 Latin American Robotics Symposium (LARS), 2021 Brazilian Symposium on Robotics (SBR), and 2021 Workshop on Robotics in Education (WRE). Presented at the 2021 Latin American Robotics Symposium (LARS), 2021 Brazilian Symposium on Robotics (SBR), and 2021 Workshop on Robotics in Education (WRE), pp. 25–29. https://doi.org/10.1109/LARS/SBR/WRE54079.2021.9605485.

Bunk, T., Varshneya, D., Vlasov, V., Nichol, A., 2020. DIET: lightweight Language Understanding for dialogue systems. ArXiv200409936 Cs. https://doi.org/10.48550/arXiv.2004.09936.

Burtsev, M., Seliverstov, A., Airapetyan, R., Arkhipov, M., Baymurzina, D., Bushkov, N., Gureenkova, O., Khakhulin, T., Kuratov, Y., Kuznetsov, D., Litinsky, A., Logacheva, V., Lymar, A., Malykh, V., Petrov, M., Polulyakh, V., Pugachev, L., Sorokin, A., Vikhreva, M., Zaynutdinov, M., 2018. DeepPavlov: open-source library for dialogue systems. In: Proceedings of ACL 2018, System Demonstrations. Association for Computational Linguistics, Melbourne, Australia, pp. 122–127. https://doi.org/10.18653/v1/P18-4021.

Carvalho, T.P., Soares, F.A.A.M.N., Vita, R., Francisco, R. da P., Basto, J.P., Alcalá, S.G.S., 2019. A systematic literature review of machine learning methods applied to predictive maintenance. Comput. Ind. Eng. 137, 106024 https://doi.org/10.1016/j.cie.2019.106024.

Chorowski, J., Bahdanau, D., Serdyuk, D., Cho, K., Bengio, Y., 2015. Attention-Based Models for Speech Recognition. https://doi.org/10.48550/arXiv.1506.07503.

Cieri, C., Miller, D., Walker, K., 2004. The Fisher corpus: a resource for the next generations of speech-to-text. In: Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04). Presented at the LREC 2004, European Language Resources Association (ELRA). Lisbon, Portugal.

Collobert, R., Puhrsch, C., Synnaeve, G., 2016. Wav2Letter: an end-to-end ConvNet-based speech recognition system. ArXiv160903193 Cs. https://doi.org/10.48550/arXiv.1609.03193.

Coucke, A., Saade, A., Ball, A., Bluche, T., Caulier, A., Leroy, D., Doumouro, C., Gisselbrecht, T., Caltagirone, F., Lavril, T., Primet, M., Dureau, J., 2018. Snips voice platform: an embedded Spoken Language understanding system for private-by-design voice interfaces. https://doi.org/10.48550/arXiv.1805.10190.

Dos Anjos, J.C.S., Matteussi, K.J., De Souza, P.R.R., Grabher, G.J.A., Borges, G.A., Barbosa, J.L.V., González, G.V., Leithardt, V.R.Q., Geyer, C.F.R., 2020. Data processing model to perform big data analytics in hybrid infrastructures. IEEE Access 8, 170281–170294. https://doi.org/10.1109/ACCESS.2020.3023344.

Falcon, W., 2015. PyTorch lightning [WWW Document]. URL https://pytorchlightning.ai/. , 12.14.21.

Font, F., Roma, G., Serra, X., 2013. Freesound technical demo. In: Proceedings of the 21st ACM International Conference on Multimedia, MM '13. Association for Computing Machinery, New York, NY, USA, pp. 411–412. https://doi.org/10.1145/2502081.2502245.

Franklin, D., 2018. NVIDIA Jetson AGX xavier delivers 32 TeraOps for new era of AI in robotics [WWW Document]. NVIDIA Tech. Blog. URL. https://developer.nvidia.com/blog/nvidia-jetson-agx-xavier-32-teraops-ai-robotics/, 2.28.22.

Ginsburg, B., Castonguay, P., Hrinchuk, O., Kuchaiev, O., Lavrukhin, V., Leary, R., Li, J., Nguyen, H., Zhang, Y., Cohen, J.M., 2020. Stochastic gradient methods with layer-wise adaptive moments for training of deep networks. ArXiv190511286 Cs Stat. https://doi.org/10.48550/arXiv.1905.11286.

Godfrey, J.J., Holliman, E.C., McDaniel, J., 1992. SWITCHBOARD: telephone speech corpus for research and development. In: Presented at the Acoustics, Speech, and Signal Processing, IEEE International Conference on. IEEE Computer Society, pp. 517–520. https://doi.org/10.1109/ICASSP.1992.225858.

Graves, A., Fernández, S., Gomez, F., Schmidhuber, J., 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In: Proceedings of the 23rd International Conference on Machine Learning, ICML '06. Association for Computing Machinery, New York, NY, USA, pp. 369–376. https://doi.org/10.1145/1143844.1143891.

Häfner, B., Bajpai, V., Ott, J., Schmitt, G.A., 2022. A survey on cooperative architectures and maneuvers for connected and automated vehicles. IEEE Commun. Surv. Tutor. 24, 380–403. https://doi.org/10.1109/COMST.2021.3138275.

Hart, M., 1971. Project Gutenberg [WWW Document]. Proj. Gutenberg. URL. https://www.gutenberg.org/, 12.14.21.

Hebbar, R., Somandepalli, K., Narayanan, S., 2019. Robust speech activity detection in movie audio: data resources and experimental evaluation. In: ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Presented at the ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 4105–4109. https://doi.org/10.1109/ICASSP.2019.8682532.

Hernández Acosta, L., Reinhardt, D., 2022. A survey on privacy issues and solutions for Voice-controlled Digital Assistants. Pervasive Mob. Comput. 80, 101523 https://doi.org/10.1016/j.pmcj.2021.101523.

Hoy, M., 2018. Alexa, Siri, Cortana, and more: an introduction to voice assistants. Med. Ref. Serv. Q. 37, 81–88. https://doi.org/10.1080/02763869.2018.1404391.

Huang, J., Kuchaiev, O., O'Neill, P., Lavrukhin, V., Li, J., Flores, A., Kucsko, G., Ginsburg, B., 2020. Cross-Language transfer learning, continuous learning, and domain adaptation for end-to-end automatic speech recognition. ArXiv200504290 Eess. https://doi.org/10.48550/arXiv.2005.04290.

Huang, J., Zhang, Y., Ginsburg, B., Chitale, P., 2019. Develop smaller speech recognition models with NVIDIA's NeMo framework [WWW Document]. NVIDIA Dev. Blog. URL. https://developer.nvidia.com/blog/develop-smaller-speech-recognition-models-with-nvidias-nemo-framework/, 2.2.22.

Huang, R., Lam, M.W.Y., Wang, J., Su, D., Yu, D., Ren, Y., Zhao, Z., 2022. FastDiff: a fast conditional diffusion model for high-quality speech synthesis. https://doi.org/10.48550/arXiv.2204.09934.

Ito, K., Johnson, L., 2017. The LJ speech dataset [WWW Document]. URL. https://keithito.com/LJ-Speech-Dataset, 6.21.23.

Jia, F., Majumdar, S., Ginsburg, B., 2021. MarbleNet: deep 1D time-channel separable convolutional neural network for voice activity detection. ArXiv201013886 Cs Eess. https://doi.org/10.48550/arXiv.2010.13886.

Jin, Z., Geng, M., Deng, J., Wang, T., Hu, S., Li, G., Liu, X., 2024. Personalized adversarial data augmentation for dysarthric and elderly speech recognition. IEEE Trans. Audio Speech Lang. Process. 32, 413–429. https://doi.org/10.1109/TASLP.2023.3323888.

Kazmi, S.M.A., Dang, T.N., Yaqoob, I., Ndikumana, A., Ahmed, E., Hussain, R., Hong, C. S., 2019. Infotainment enabled smart cars: a joint communication, caching, and computation approach. IEEE Trans. Veh. Technol. 68, 8408–8420. https://doi.org/10.1109/TVT.2019.2930601.

Koh, J., Mislan, A., Khoo, K., Ang, B., Ang, W., Ng, C., Tan, Y.-Y., 2019. Building the Singapore English National Speech Corpus. https://doi.org/10.21437/Interspeech.2019-1525.

Kong, J., Kim, J., Bae, J., 2020. HiFi-GAN: generative adversarial networks for efficient and high fidelity speech synthesis. ArXiv201005646 Cs Eess. https://doi.org/10.48550/arXiv.2010.05646.

Kriman, S., Beliaev, S., Ginsburg, B., Huang, J., Kuchaiev, O., Lavrukhin, V., Leary, R., Li, J., Zhang, Y., 2020. Quartznet: deep automatic speech recognition with 1D time-channel separable convolutions. In: ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Presented at the ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 6124–6128. https://doi.org/10.1109/ICASSP40776.2020.9053889.

Kuchaiev, O., Li, J., Nguyen, H., Hrinchuk, O., Leary, R., Ginsburg, B., Kriman, S., Beliaev, S., Lavrukhin, V., Cook, J., Castonguay, P., Popova, M., Huang, J., Cohen, J. M., 2019. NeMo: a toolkit for building AI applications using Neural Modules. ArXiv190909577 Cs Eess. https://doi.org/10.48550/arXiv.1909.09577.

Kumar, K., Kumar, R., de Boissiere, T., Gestin, L., Teoh, W.Z., Sotelo, J., de Brebisson, A., Bengio, Y., Courville, A., 2019. MelGAN: generative adversarial networks for conditional waveform synthesis. ArXiv191006711 Cs Eess. https://doi.org/10.48550/arXiv.1910.06711.

Łańcucki, A., 2021. Fastpitch: parallel text-to-speech with pitch prediction. In: ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Presented at the ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 6588–6592. https://doi.org/10.1109/ICASSP39728.2021.9413889.

Li, W., Cao, D., Tan, R., Shi, T., Gao, Z., Ma, J., Guo, G., Hu, H., Feng, J., Wang, L., 2024. Intelligent cockpit for intelligent connected vehicles: definition, taxonomy, technology and evaluation. IEEE Trans. Intell. Veh. 9, 3140–3153. https://doi.org/10.1109/TIV.2023.3339798.

Liu, H., Liu, J., Cui, L., Teng, Z., Duan, N., Zhou, M., Zhang, Y., 2023. LogiQA 2.0—an improved dataset for logical reasoning in Natural Language Understanding. IEEE Trans. Audio Speech Lang. Process. 31, 2947–2962. https://doi.org/10.1109/TASLP.2023.3293046.

Liu, J., Wan, F., Zou, J., Zhang, J., 2023. Exploring factors affecting people's willingness to use a voice-based in-car assistant in electric cars: an empirical study. World Electr. Veh. J. 14, 73. https://doi.org/10.3390/wevj14030073.

Liu, R., Sisman, B., Gao, G., Li, H., 2024. Controllable accented text-to-speech synthesis with fine and coarse-grained intensity rendering. IEEE Trans. Audio Speech Lang. Process. 32, 2188–2201. https://doi.org/10.1109/TASLP.2024.3378110.

Loshchilov, I., Hutter, F., 2017. SGDR: stochastic gradient descent with warm restarts. ArXiv160803983 Cs Math. https://doi.org/10.48550/arXiv.1608.03983.

Lugosch, L., Ravanelli, M., Ignoto, P., Tomar, V.S., Bengio, Y., 2019. Speech model pre-training for end-to-end Spoken Language understanding. ArXiv190403670 Cs Eess. https://doi.org/10.48550/arXiv.1904.03670.

Lund, W., Kennard, D., Ringger, E., 2013. Combining multiple thresholding binarization values to improve OCR output. https://doi.org/10.1117/12.2006228.

Mahmood, A., Wang, J., Yao, B., Wang, D., Huang, C.-M., 2023. LLM-powered conversational voice assistants: interaction patterns, opportunities, challenges, and design guidelines. https://doi.org/10.48550/arXiv.2309.13879.

Massai, L., Nesi, P., Pantaleo, G., 2019. PAVAL: a location-aware virtual personal assistant for retrieving geolocated points of interest and location-based services. Eng. Appl. Artif. Intell. 77, 70–85. https://doi.org/10.1016/j.engappai.2018.09.013.

McGuire, H., 2014. LibriVox: free public domain audiobooks. Ref. Rev. 28, 7–8. https://doi.org/10.1108/RR-08-2013-0197.

Mitrevski, M., 2018. Getting started with Wit.ai. In: Mitrevski, M. (Ed.), Developing Conversational Interfaces for iOS: Add Responsive Voice Control to Your Apps. Apress, Berkeley, CA, pp. 143–164. https://doi.org/10.1007/978-1-4842-3396-2_5.

Mittal, S., 2019. A Survey on optimized implementation of deep learning models on the NVIDIA Jetson platform. J. Syst. Architect. 97, 428–442. https://doi.org/10.1016/j.sysarc.2019.01.011.
Mozilla, 2022. Project DeepSpeech [WWW Document]. URL. https://github.com/mozilla/DeepSpeech. (Accessed 2 February 2022).
Mozilla, 2020. Common voice [WWW Document]. URL. https://commonvoice.mozilla.org/, 12.14.21.
Munteanu, C., Penn, G., Baecker, R., Toms, E., James, D., 2006. Measuring the acceptable word error rate of machine-generated webcast transcripts. In: Ninth International Conference on Spoken Language Processing. Citeseer.
Nagari, N., 2020. Comparing 4 popular open source speech to text neural network models. Medium. URL. https://medium.com/nick.nagari/comparing-4-popular-open-source-speech-to-text-neural-network-models-92676a9f9265, 2.2.22.
Nekvinda, T., Dušek, O., 2020. One model, many languages: meta-learning for multilingual text-to-speech. ArXiv200800768 Cs eess. https://doi.org/10.48550/arXiv.2008.00768.
NVIDIA, 2023a. TTS vocoder uniglow [WWW Document]. URL https://catalog.ngc.nvidia.com/orgs/nvidia/teams/nemo/models/tts_uniglow. .
NVIDIA, 2023b. TTS en E2E FastPitch hifigan | NVIDIA NGC [WWW Document]. NGC Cat. URL. https://catalog.ngc.nvidia.com/orgs/nvidia/teams/nemo/models/tts_en_e2e_fastpitchhifigan. (Accessed 2 February 2022).
NVIDIA, 2023c. TTS en E2E Fastspeech2 hifigan NVIDIA NGC [WWW Document]. NGC Cat. URL. https://catalog.ngc.nvidia.com/orgs/nvidia/teams/nemo/models/tts_en_e2e_fastspeech2hifigan. (Accessed 2 February 2022).
NVIDIA, 2023d. Multidataset-QuartzNet15x5 | NVIDIA NGC [WWW Document]. URL. https://catalog.ngc.nvidia.com/orgs/nvidia/models/multidataset_quartznet15x5, 12.14.21.
NVIDIA, 2018. apex.amp — apex 0.1.0 documentation [WWW Document]. URL. https://nvidia.github.io/apex/amp.html.
Okumura, H., 2019. Human centric AR amp;VR display and interface technologies for automobile. IEEE Consum. Electron. Mag. 8, 60–61. https://doi.org/10.1109/MCE.2019.2923900.
Panayotov, V., Chen, G., Povey, D., Khudanpur, S., 2015. Librispeech: an ASR corpus based on public domain audio books. In: 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Presented at the 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 5206–5210. https://doi.org/10.1109/ICASSP.2015.7178964.
Paul, D.B., Baker, J.M., 1992. The design for the wall street journal-based CSR corpus. In: Speech and Natural Language: Proceedings of a Workshop Held at Harriman, New York, February 23-26, 1992. Presented at the HLT 1992.
Perera, V., Chung, T., Kollar, T., Strubell, E., 2018. Multi-task learning for parsing the alexa meaning representation language. In: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence, AAAI'18/IAAI'18/EAAI'18. AAAI Press, New Orleans, Louisiana, USA, pp. 5390–5397. https://doi.org/10.1609/aaai.v32i1.12019.
Pham, H., 2006. Pyaudio: portaudio v19 python bindings [WWW Document]. URL. https://people.csail.mit.edu/hubert/pyaudio/docs/, 2.28.22.
Picovoice, 2022a. Leopard [WWW Document]. URL. https://github.com/Picovoice/leopard, 4.19.22.
Picovoice, 2022b. Speech-to-Text benchmark [WWW Document]. URL. https://github.com/Picovoice/speech-to-text-benchmark, 4.27.22.
Polyakov, E.V., Mazhanov, M.S., Rolich, A.Y., Voskov, L.S., Kachalova, M.V., Polyakov, S.V., 2018. Investigation and development of the intelligent voice assistant for the Internet of Things using machine learning. In: 2018 Moscow Workshop on Electronic and Networking Technologies (MWENT), pp. 1–5. https://doi.org/10.1109/MWENT.2018.8337236. Presented at the 2018 Moscow Workshop on Electronic and Networking Technologies (MWENT).
The kaldi speech recognition toolkit. In: Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlicek, P., Qian, Y., Schwarz, P., Silovsky, J., Stemmer, G., Vesely, K. (Eds.), 2011. Presented at the IEEE 2011 Workshop on Automatic Speech Recognition and Understanding. IEEE Signal Processing Society.
Pratap, V., Xu, Q., Sriram, A., Synnaeve, G., Collobert, R., 2020. MLS: a large-scale multilingual dataset for speech research. Interspeech 2020, 2757–2761. https://doi.org/10.21437/Interspeech.2020-2826.
Prechelt, L., 1998. Early stopping - but when? In: Orr, G.B., Müller, K.-R. (Eds.), Neural Networks: Tricks of the Trade, Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, pp. 55–69. https://doi.org/10.1007/3-540-49430-8_3.
Prenger, R., Valle, R., Catanzaro, B., 2019. Waveglow: a flow-based generative network for speech synthesis. In: ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Presented at the ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 3617–3621. https://doi.org/10.1109/ICASSP.2019.8683143.
Rasa, 2024. Rasa docs [WWW document]. URL. https://rasa.com/docs/rasa/tuning-your-model/, 2.25.22.
Ren, Y., Hu, C., Tan, X., Qin, T., Zhao, S., Zhao, Z., Liu, T.-Y., 2021. FastSpeech 2: fast and high-quality end-to-end text to speech. ArXiv200604558 Cs Eess. https://doi.org/10.48550/arXiv.2006.04558.
Rhode, S., Van Vaerenbergh, S., Pfriem, M., 2020. Power prediction for electric vehicles using online machine learning. Eng. Appl. Artif. Intell. 87, 103278 https://doi.org/10.1016/j.engappai.2019.103278.
Sabharwal, N., Agrawal, A., 2020. Cognitive Virtual Assistants Using Google Dialogflow: Develop Complex Cognitive Bots Using the Google Dialogflow Platform. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-5741-8.
Savoir-faire Linux, 2022. num2words library - convert numbers to words in multiple languages [WWW Document]. URL https://github.com/savoirfairelinux/num2words. , 2.28.22.
Saxon, M., Choudhary, S., McKenna, J.P., Mouchtaris, A., 2021. End-to-End Spoken Language Understanding for Generalized Voice Assistants, vol. 2021. Interspeech, pp. 4738–4742. https://doi.org/10.21437/Interspeech.2021-1826.
Seymour, W., Zhan, X., Coté, M., Such, J., 2023. A systematic review of ethical concerns with voice assistants. In: Proceedings of the 2023 AAAI/ACM Conference on AI, Ethics, and Society, AIES '23. Association for Computing Machinery, New York, NY, USA, pp. 131–145. https://doi.org/10.1145/3600211.3604679.
Shafeeg, A., Shazhaev, I., Mihaylov, D., Tularov, A., Shazhaev, I., 2023. Voice assistant integrated with chat GPT. Indones. J. Comput. Sci. 12 https://doi.org/10.33022/ijcs.v12i1.3146.
Shen, J., Pang, R., Weiss, R.J., Schuster, M., Jaitly, N., Yang, Z., Chen, Z., Zhang, Y., Wang, Y., Skerry-Ryan, R.J., Saurous, R.A., Agiomyrgiannakis, Y., Wu, Y., 2018. Natural TTS synthesis by conditioning WaveNet on mel spectrogram predictions. ArXiv171205884 Cs. https://doi.org/10.48550/arXiv.1712.05884.
Shi, W., Cao, J., Zhang, Q., Li, Y., Xu, L., 2016. Edge computing: vision and challenges. IEEE Internet Things J. 3, 637–646. https://doi.org/10.1109/JIOT.2016.2579198.
Solak, I., 2019. The M-AILABS speech dataset. URL. https://www.caito.de/2019/01/03/the-m-ailabs-speech-dataset/, 6.20.24.
Son, D.B., Binh, T.H., Vo, H.K., Nguyen, B.M., Binh, H.T.T., Yu, S., 2022. Value-based reinforcement learning approaches for task offloading in delay constrained vehicular edge computing. Eng. Appl. Artif. Intell. 113, 104898 https://doi.org/10.1016/j.engappai.2022.104898.
Tan, X., Chen, J., Liu, H., Cong, J., Zhang, C., Liu, Y., Wang, X., Leng, Y., Yi, Y., He, L., Zhao, S., Qin, T., Soong, F., Liu, T.-Y., 2024. NaturalSpeech: end-to-end text-to-speech synthesis with human-level quality. IEEE Trans. Pattern Anal. Mach. Intell. 46, 4234–4245. https://doi.org/10.1109/TPAMI.2024.3356232.
Tekur, C., Gardiner, F., 2019. Nvidia {"NGC"} Deep Learning Containers. USENIX Association, Santa Clara, CA.
Torrey, L., Shavlik, J., 2010. Transfer learning. IGI Global. https://doi.org/10.4018/978-1-60566-766-9.ch011.
Urban, E. and Mehrotra, N., 2023. Test accuracy of a Custom Speech model [WWW Document]. URL https://learn.microsoft.com/en-us/azure/cognitive-services/speech-service/how-to-custom-speech-evaluate-data 6.18.23.
Vaessen, N., 2022. JiWER: similarity measures for automatic speech recognition evaluation [WWW Document]. URL https://github.com/jitsi/jiwer. , 2.16.22.
Valle, R., Puri, R., Tabuov, R., Anisimov, I., Sîng-hông, S., 2024. NVIDIA/tacotron2 [WWW Document]. URL. https://github.com/NVIDIA/tacotron2, 6.20.24.
Vasiliev, Y., 2020. Natural Language Processing with Python and spaCy: A Practical Introduction. No Starch Press.
Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I., 2017. Attention is all you need. ArXiv170603762 Cs. https://doi.org/10.48550/arXiv.1706.03762.
Vu, M.D., Wang, H., Li, Z., Chen, J., Zhao, S., Xing, Z., Chen, C., 2024. GPTVoiceTasker: LLM-powered virtual assistant for smartphone. https://doi.org/10.48550/arXiv.2401.14268.
Wang, D., Zheng, T.F., 2015. Transfer learning for speech and language processing. In: 2015 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA), pp. 1225–1237. https://doi.org/10.1109/APSIPA.2015.7415532. Presented at the 2015 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA).
Wang, X., Wei, T., Kong, L., He, L., Wu, F., Chen, G., 2019. ECASS: edge computing based auxiliary sensing system for self-driving vehicles. J. Syst. Architect. 97, 258–268. https://doi.org/10.1016/j.sysarc.2019.02.014.
Weng, Z., Qin, Z., Tao, X., Pan, C., Liu, G., Li, G.Y., 2023. Deep learning enabled semantic communications with speech recognition and synthesis. https://doi.org/10.48550/arXiv.2205.04603.
Williams, S., 2018. Hands-on Chatbot Development with Alexa Skills and Amazon Lex : Create Custom Conversational and Voice Interfaces for Your Amazon Echo Devices and Web Platforms. Packt, Birmingham.
Xiao, R., Wan, Y., Yang, B., Zhang, H., Tang, H., Wong, D.F., Chen, B., 2023. Towards energy-preserving Natural Language Understanding with spiking neural networks. IEEE Trans. Audio Speech Lang. Process. 31, 439–447. https://doi.org/10.1109/TASLP.2022.3221011.
Xiong, W., Droppo, J., Huang, X., Seide, F., Seltzer, M.L., Stolcke, A., Yu, D., Zweig, G., 2017. Toward human parity in conversational speech recognition. IEEE Trans. Audio Speech Lang. Process. 25, 2410–2423. https://doi.org/10.1109/TASLP.2017.2756440.
Zhai, B., Gao, T., Xue, F., Rothchild, D., Wu, B., Gonzalez, J.E., Keutzer, K., 2020. SqueezeWave: extremely lightweight vocoders for on-device speech synthesis. ArXiv200105685 Cs Eess. https://doi.org/10.48550/arXiv.2001.05685.
Zhou, X., Zhang, M., Zhou, Y., Wu, Z., Li, H., 2024. Accented text-to-speech synthesis with limited data. IEEE Trans. Audio Speech Lang. Process. 32, 1699–1711. https://doi.org/10.1109/TASLP.2024.3363414.