

Arabic Text-to-Speech Based on ESPnet Platform

Ali Habeeb Hasan, Dr. Sabrina Tiun

Faculty of Information Science and Technology
University Kebangsaan Malaysia
43600 UKM Bangi, Selangor Malaysia

P103916@siswa.ukm.edu.my it.ali.habeeb.hasan@gmail.com

Abstract

TTS (Text-to-Speech) represents one of the crucial factors for human and machine communication and interaction. Therefore, this technique transforms written text into vocalized audio and thus, an instrument like a robot can communicate by using speech with its surroundings. Substantially, the TTS system consists of two phases, First, the Text-processing phase in which the input text is converted into a phonetic characterization alongside the optional meta-data such as stress tags. The second phase represents the creation of an audio waveform throughout the phonetic presentations.

The main objective of this study is to create an End-to-End Speech Processing Toolkit (ESPnet) that supports the Arabic languages. In the Arabic language, ESPnet is a brand-new project that combines prosperous automatic speech recognition-related basics, frameworks, and systems to encourage and verify the suggested front-end execution, for instance, speech development and segregation. There are a few applications that support TTS for the Arabic language that is not flexible like ESPnet. Therefore, it is necessary to reinforce the Arabic language throughout ESPnet to support various applications that provide text-to-speech for the Arabic language. All-in-one recipes will be provided embracing data pre-processing, factor extraction, training, and evaluation channels for a broad range of standard databases. This study displays the plan of the toolkit, some important functionalities, specifically the speech

recognition integration that separates ESPnet from further open-source toolsets, and experimental results via principal benchmark datasets.

LIST OF ABBREVIATIONS: *One-Dimensional, Artificial Intelligence, Application Programming Interface, Automatic Speech Recognition, Convolutional Neural Network, 2-Dimensional Convolutional Layers, Conditional Random Fields, Connectionist Temporal Classification, Deep Learning, Deep Neural Network, End-to-End TTS, End-to-End Speech Processing Toolkit, Grapheme-to-phoneme, Gated Recurrent Unit, Graphical User Interface, Hidden Conditional Random Field, Markov Model, Hyper Text Transfer Protocol, Long Short-Term Memory, Letter-To-Sound, Mean Opinion Score, Machine Translation, Natural Language Processing, Neural Network, Non-Sequential Greedy Decoding, Nonstandard Words, Out Of Vocabulary, Phoneme Error Rate, Part-of-Speech, Recurrent Neural Network, Speech Development and Separation, Speech Language Understanding, Speech Translation, Text-to-Speech, Universiti Kebangsaan Malaysia, Unified Resource Link, Word Error Rate, Weighted Finite-State Transduce*

1 INTRODUCTION

Synthetic voice is provided by Text-to-Speech (TTS) technology through textual data only. Hence, it serves in human-machine communication as an additional natural interface. TTS can help humans in various applications such as educational and medical applications. Overall, modern TTS is based on statistical parametric and unit selection strategies. Specific observation has been granted to Deep Neural Network (DNN)-based TTS recently because of its benefits in terms of flexibility and robustness.

There are two main stages for TTS to be achieved: Text Normalization and Grapheme-to-phoneme (G2P) Conversion. G2P conversion is mainly the chore of forecasting the pronunciation of a word given in its written or graphemic structure. It consists of a crucial fragment of both TTS and automatic speech recognition (ASR) systems. The G2P framework's quality has a huge impact on the quality of speech. Flawed G2P conversion leads to an abnormal pronunciation or even obscure synthetic talk.

TTS platforms require working with texts that include non-standard terms, followed by numbers, currency, dates, and abbreviations. Based on this, text normalization is the chore function for a TTS setup to transform written-form contents into spoken-form strings. ESPnet has become a developed technology primarily in speech processing sections through open-source applications that encourage both TTS and ASR. Furthermore, the new open-source End-to-end speech processing toolkit intends to offer an endless neural end-to-end system for speech proceeding.

ESPnet offers a completed architecture for speech recognition and several speech processing examinations through using chainer and Pytorch as a deep learning engine. Also, it offers Kaldi-style data processing, feature extraction/format, and recipes. Generally, some of the fundamental features of the Kaldi style process for TTS and ASR.

The transducer based on end-to-end processing is the chore of **ESPnet**. Recurrent Neural Network-based (RNN-based) encoder and decoder consist of one of the main structures of ESPnet. RNNs master the sequential characteristics of information and implement techniques to forecast the next situation. They are used in extensive acquiring and in the enhancement of frameworks that affect the mobility of neurons in the human brain. RNN applies feedback loops to implement a sequence of information that shapes the result. Such feedback loops let the data be stored for a certain period. This influence is usually identified as memory. RNN uses cases that aim to relate to linguistic patterns where the succeeding letter in a word or the following word in a sentence is anticipated via the information

preceding it. Long short-term memory (LSTM) networks implement standard and private units which are considered a type of RNN. LSTM units involve a "memory cell" that stores data in memory for long periods. This type of RNN can be taught in different languages, such as the Arabic language that we intend to add to the ESPnet.

2. LITERATURE

2.1 TTS and ESPnet

TTS is considered as an additional natural interface between humans and machines communication. Modern TTS is based on statistical parametric and unit selection strategies. Specific observation has been granted to DNN-based TTS recently because of its benefits in terms of flexibility and robustness. TTS systems involve two sub-systems: speech generation and NLP. TTS platforms require working with texts that include non-standard terms, followed by numbers, currency, dates, and abbreviations.

ESPnet has become a developed technology primarily in speech processing sections through open-source applications that encourage both TTS and ASR. Furthermore, the new open-source End-to-end speech processing toolkit intends to offer an endless neural end-to-end system for speech proceeding. ESPnet provides various well-known techniques such as RNN, TDNN, and many others... ESPnet offers a completed architecture for speech recognition and several speech processing examinations through using chainer as a deep learning engine.

DL is the heart of ESPnet. It is the method used to train ESPnet for new languages, such as the Arabic language in our case. DL is represented through the RNN network. Through training the RNN we are changing its weights to give us the expected result. Each iteration of data in the RNN will get results closer to expected. Thus, there will be a stage where the error rate of the network is negligible. The input data for this RNN is wave sound data processed in an understandable way for the network. However, the training output is the sentence related to the wave data.

This Literature review lays out the required attributes, characteristics, and strategies of TTS for this thesis. It consists of a deep review in which identical fields have been added also.

First, a general introduction is displayed followed by the synthesizer technologies in section 2.2. Second, deep learning methods are explained in section 2.3 followed by Recurrent Neural Networks (RNNs) in

section 2.4. Long-Short Term Memory (LSTM) and Bidirectional Long-Short Term Memory (Bi-LSTM) represent the subsections of RNNs examined thoroughly. Then, section 2.5 analyzes the Convolutional Neural Networks, an exclusive type of neural network for developing information. End-to-end training is examined in section 2.6 to merge various components in the computational graph of the neural network and maximize it. Finally, Tacotron which is an end-to-end generative TTS model was explained. Finally, the Attention Layer is represented.

2.2 SYNTHESIZER TECHNOLOGIES

Speech Synthesis is the presentation of human speech by a computer. For several years, an automatic generation of speech wave types has been developed. New developments in speech synthesis have been created with increased intelligibility, but the naturalness and the quality of the sound remain a challenging issue. Several issues and challenges in Speech Synthesis exist such as Text to phoneme, text normalization, etc.

2.2.1. TEXT NORMALIZATION

In speech and language technologies, including its various types of tasks, text normalization is one of the key parts. Generally, text normalization frameworks transform a dictated representation of a text into a type of how that context should be read aloud.

Recently, personal assistant applications offered appropriate responses based on customer inquiries through NLP techniques. The primary stage for these applications is the normalization of the query. Several previous works used mainly custom-built rules where such rules are amended to specific domains of these platforms. Also, weighted finite-state transducers (WFSTs) were used.

The language information contains a series of data like streams of characters and arrangement of words. These characteristics affect the NLP techniques where arithmetic models can easily interact with information. Recently, deep learning has accomplished a futuristic performance in various NLP-associated domains. Consequently, additional information is needed to use deep learning techniques. Nevertheless, to train and operate such techniques requires reduced linguistic expertise.

Lately, for text normalization, double encoder arrangements were examined. They are called Siamese models which involve a set of encoders that encrypt pairs of inputs into vectors, and a framework to

compare the resemblance between the two inputs. Such a model necessitates training of both negative and positive information, and it was employed from the previous training and test data of the 2017 Kaggle competition on text normalization.

Furthermore, the contextual seq2seq model was suggested, which utilizes a sliding window and RNN. In this framework, bi-directional GRU is implemented for both the encoder and the decoder, and the text words are categorized with “<self>”, leading the model to differentiate the nonstandard words (NSW) and the text. To rectify the results of a connectionist temporal categorization method in Mandarin ASR, a transformer-based phonetic correction framework is presented.

2.2.2. Grapheme-to-Phoneme Conversion

The phonetic transcription has been generated from the written type of the words because of the procedure for grapheme-to-phoneme transformation. The phonetic type of the word is called phoneme sequence (or phonemes), whereas the spelling form of the word is called grapheme sequence (or graphemes). In ASR platforms and TTS, it is crucial to enhance phonemic representation. Pronunciation vocabulary lists represent the middle layer among acoustic and language frameworks. For an advanced speech identification task, the overall platform of the performance is based on the quality of the pronunciation element. Particularly, the framework’s performance is based on G2P efficiency.

During a long period, G2P alteration has been examined. Rule-based G2P frameworks utilize a wide range of grapheme-to-phoneme controls. Linguistic expertise is the fundamental step in expanding such a G2P model. Primary grapheme-phoneme sequence conjunction has been established through such frameworks to calculate a combined n-gram language system over the chain. Afterward, Hidden Conditional Random Field (HCRF) strategy was created where the arrangement between grapheme and phoneme progression is shaped through unrevealed features. In general, the HCRF representation provides a significant competitive outcome, but training such a model requires a challenging memory and intensive computation.

In G2P conversion also, neural networks have been applied. They consist of robust with regards to spelling mistakes and OOV words where they generalize very well. Besides, they can be smoothly combined into end-to-end TTS/ASR platforms. A TTS structure (Deep Voice) is established, which was built totally from DNN. Deep Voice is the main step to truthfully end-to-end neural speech combination. Therefore, the whole quality of the platform is increased through the G2P system that is collectively trained with additional crucial elements of the speech synthesizer and recognizer.

Through sequence-to-sequence learning, the decoding phase is generally based on sequences, from left to right one step at a time, and the results from the preceding stages are implemented as decoder data. Based on the task and the system, sequential decoding can negatively impact the outcomes. For G2P, the non-sequential greedy decoding (NSGD) strategy was examined and merged through a complete convolutional encoder-decoder structure.

Lately, to improve the correctness through distilling the comprehension resulting from further unlabeled information and decreasing the framework size, thus sustaining the maximum accuracy, a token-level collection distillation for G2P transformation was suggested. A conversion strategy was utilized to increase the correctness of G2P transformation furthermore. In addition, the DNN-based G2P rectifier leads to complete well in languages through irregular pronunciation followed by regular pronunciation languages that can be efficiently obvious through various transcription rules.

2.3 Deep Learning Methods

To build a strong deep learning strategy, several methods are employed encompassing adaptive learning rate, dropout, batch normalization, etc... The following consist of the key aspects of these frameworks.

Adaptive learning rate: This method consists of altering the learning rate to enhance performance and decrease the training time.

Dropout: The dropout strategy is another step to cope with overfitting in thorough neural networks. During training, this strategy is used through random dropping components along with the necessary parameters in deep neural channels.

Batch normalization: To normalize activations in the middle layers of DNN, the batch normalization technique is used. It leads to accelerate training and makes learning easier. Through an applied modern image classification framework, batch Normalization accomplishes the same accuracy by 14 times.

Residual connections: To build identity mapping, residual communications and blocks, components are created through a set of collected layers, by adding the inputs to their outputs.

Transfer learning: During transfer learning, a framework trained on a specific task is utilized on another connected task. The acquired knowledge while solving a specific issue can be moved to another channel, which consists of further training on an associated issue. In various scenarios, there is a lack of

information to train the models. To start with a pre-trained framework can lead to reach greater outcomes, whereas training a framework from the beginning through insufficient data might result in reduced performance.

Max-Pooling: A filter is predetermined in a max-pooling, and this filter is utilized over the sub-areas of the input by taking its highest values. Through max-pooling, dimensions and the computational process can be lower.

2.4 Recurrent Neural Networks (RNNs)

In several NLP tasks, recurrent neural networks (RNNs) demonstrated significant outcomes. Through time-series and sequential databases, they are able of learning attributes and long-term reliance. RNNs and their alternatives; Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) have shown promising progress in several NLP tasks.

2.4.1 Long-Short Term Memory (LSTM)

Long Short-Term Memory networks (LSTM) represent one of the exclusive types of RNN, accomplish a learning long-term reliance. This module encompasses a unique memory cell that can save previous data. An LSTM unit grabs as input its past cell and unrevealed states and outputs its recent cell and concealed states. More precisely, the LSTM unit is formed of 4 gates, working in an appropriate way.

The following consist of the gates of an LSTM unit:

- input gate layer: $i_t = \sigma(W_{ixt} + U_{iht-1} + bi)$
- forget gate layer: $f_t = \sigma(W_{fxt} + U_{fht-1} + bf)$
- output candidate layer: $O_t = \sigma(W_{oxt} + U_{oht-1} + bo)$
- cell state candidate layer: $\tilde{c}_t = \tanh(W_{cxt} + U_{cht-1} + bc)$

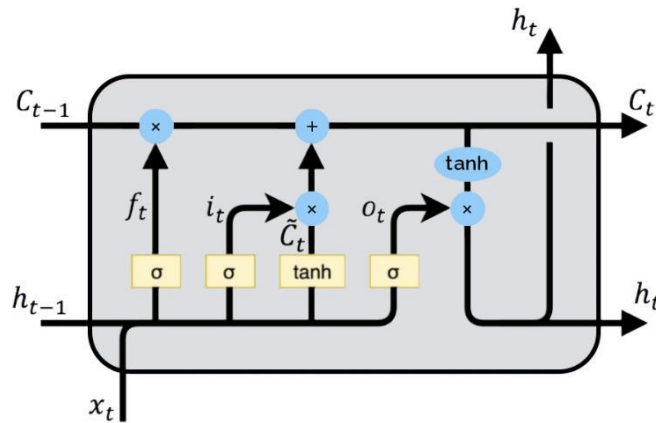


Figure 1 A basic representation of the LSTM cell

The input gate i_t identifies the level to which the input data is added to the memory.

The forget gate f_t defines the degree to which the current memory is forgotten.

The output gate of O_t of every LSTM unit at time t is calculated to receive the output memory.

Subsequently, the data in the memory cell is upgraded via half forgetting of the database gathered in the memory cell $ct-1$ and prepared throughout \tilde{c}_t . And finally, the output concealed state h_t is updated through the calculated cell state ct . LSTM successfully escapes the fading gradient issue while comparing with the basic RNN through initiating the gate process, which is beneficial in coping with long-term dependencies.

2.4.2 Bidirectional Long-Short Term Memory (Bi-LSTM)

Bidirectional LSTM (Bi-LSTM) operates input series in both directions via 2 sub-layers to consider the whole input conditions. For each of the time steps, these 2 sub-layers calculate the future hidden sequence h^{\rightarrow} and the previously hidden sequence h^{\leftarrow} , for every time step. The arrow that points in both directions (right and left) represents the backward layer.

One disadvantage of Bi-LSTM is that before it makes predictions, the whole series should be available. In various applications like real-time speech recognition, the whole announcement may not be available, therefore, the Bi-LSTM is not accurate. However, for various NLP applications where at the same time the whole phrase is available, the standard Bi-LSTM design is efficient. Furthermore, Bi-LSTM is slower than LSTM because the outcomes of the onwards pass must be accessible for the previous pass to continue.

that points to the right represents the forward layer, however, the arrow that points in both directions (right and left) represents the backward layer.

2.5 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) represent an exclusive type of neural network for developing data that has a transitory or conceptual relationship. CNN's are involved in several domains such as ("image, object and handwriting recognition, face verification, machine translation, speech synthesis"). The construction of vanilla CNN is built on various layer types where every layer brings out a particular function as displayed in Fig 4.

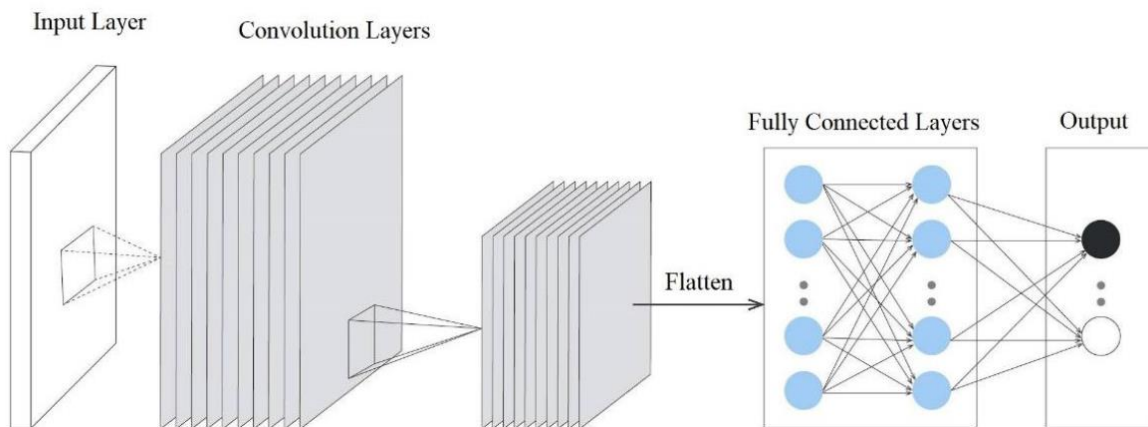


Figure 2 The general architecture of CNN

The basic component of the CNN design is the convolution layer that accelerates via the input with one or more filters, executing a complexity operation between every input field and the filter. The outcomes will be gathered in activation maps that are also called feature maps, which are the convolution layer output. Crucially, activation maps can encompass characteristics that several kernels did bring out.

After a convolutional layer, a pooling layer is often used. The primary benefit of applying the pooling strategy is that it exceptionally decreases the number of trainable dimensions and initiates translation invariance. The most frequent way to do pooling is to implement a maximal operation to the outcomes of each filter. Nevertheless, there are several types of pooling methods such as stochastic pooling, fractional max-pooling, and averaging pooling.

Deep learning strategy includes various developing types of CNN, such as "1D-CNN, 2D-CNN, and 3D-CNN". Various tasks that employ images or videos as inputs represent a more similar use of 2D-CNNs

than 1D-CNNs and 3D-CNNs. CNN accomplish a very successful representation learning where it consists of the primary reasons to make convolutional neural networks superior to past methods. It also analyzes spatial and temporal connections. and modeling together. Therefore, the model turns spectrally tolerant: common representations are acquired in various areas of the input, and the whole number of dimensions can be crucially lowered.

2.6 End-to-end training

Increased accuracy in several application domains can be achieved, encompassing NLP through end-to-end training of deep learning frameworks via a large database. The main goal of end-to-end training is to merge various elements in the computational graph of the neural network by maximizing it as a total.

In several tasks, end-to-end solutions have reached significant outcomes. This end-to-end adversarial TTS works on either pure text or raw, such as temporally inactive phoneme input sequences, and performs raw speech waveforms as output. These frameworks remove the common intermediate bottlenecks available in most modern TTS engines through sustaining learned intermediate feature representations via the network. A recent TTS framework, called Tacotron, was introduced, an end-to-end generative TTS standard that synthesizes speech automatically from characters. The design of Tacotron is developed through integrating a normalizing flow into the autoregressive decoder. In other words, they implemented a text normalization pipeline to map input text into a series of phones.

2.6 Tacotron

Usually, a TTS synthesis system contains multiple layers, for instance, a text analysis frontend, an acoustic framework, and an audio synthesis model. Establishing such elements usually needs broad domain expertise and might include brittle design options. Researchers worked on an end-to-end generative TTS model, Tacotron, that arranges speech automatically from attributes through a seq2seq model with attention. Figure 3 represents the model, which encompasses an encoder, an attention-based decoder, and a post-processing net. Given <text, audio> sets, the system can be trained totally from the beginning with arbitrary initialization. Furthermore, since Tacotron provides speech at the frame level, it's significantly quicker than sample-level autoregressive strategies. Tacotron generates a "3.82" mean opinion score (MOS) on a US English eval set through a simple waveform synthesis tool.

State-of-the-art TTS pipelines are complicated. For instance, it is common for statistical parametric TTS to have a text frontend extracting several linguistic characteristics, a duration framework, an acoustic attribute prediction model, and a detailed signal-processing-based vocoder. These elements depend on extensive field expertise and are challenging to compose. Further, they are trained solely, therefore, errors from every component may mix. The difficulty of futuristic TTS architects hence guides to considerable engineering achievements when creating a novel framework.

Hence, there are several benefits of an integrated end-to-end TTS platform that can be trained on <text, audio> pairs through lower human interaction. Initially, such a system reduces the necessity for active attribute engineering, that may include heuristics and brittle design options. Next, it smoothly permits for wealthy conditioning on several characteristics, for instance, language or high-level attributes such as emotion. This is due to the conditioning that can perform at the very start of the framework instead of solely on various elements. Equivalently, alteration to recent information might also be effortless. Ultimately, a unique framework is probably to be stronger rather than a multi-stage platform where each element's mistake can compound. These benefits mean that an end-to-end framework could help to train on great amounts of developed, expressive but frequently noisy data initiated in the real world.

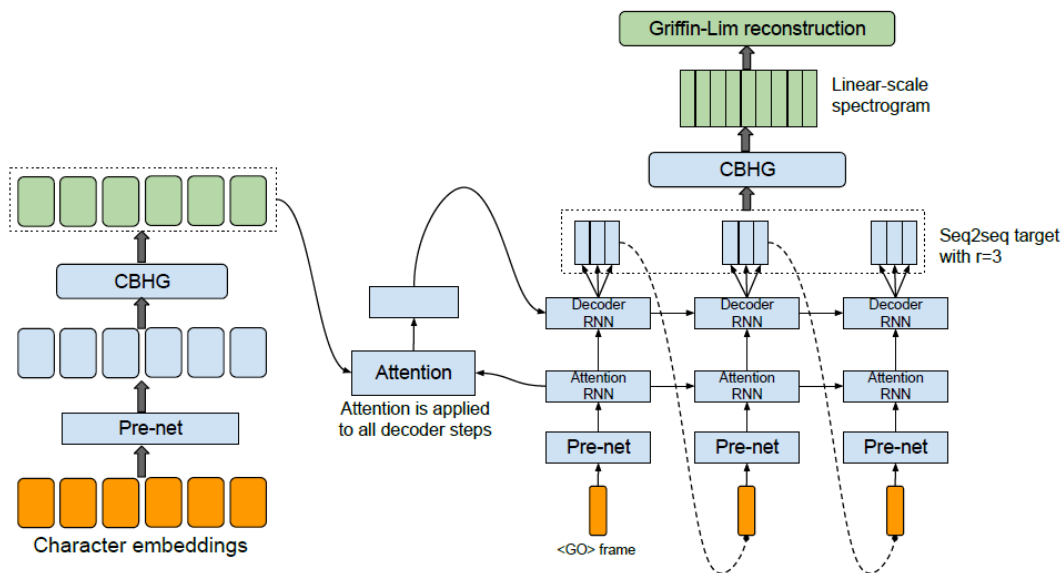


Figure 3 Model design. The framework takes attributes as input and outputs the matching raw spectrogram, which is further fed to the Griffin-Lim rebuilding algorithm to synthesize speech.

As far as we know, the earliest work that includes end-to-end TTS is through implementing seq2seq with attention. However, it needs a pre-trained hidden Markov model (HMM) aligner to assist the seq2seq model to learn the alignment. Besides, a few techniques are implemented to get the model trained, which the author's note hurts prosody. Plus, it forecasts vocoder dimensions and thus, needs a vocoder. In addition, the model is trained on phoneme inputs and the experimental outcomes appear to be limited.

2.7 Attention Layer

The fundamental element of complex recurrent or convolutional neural networks, that involve an encoder and a decoder, is the vital sequence transduction frameworks. The significant performing representations also relate to the encoder and decoder via an attention structure.

Recurrent frameworks usually lead to computation between the symbol positions of the input and output series. Associating the positions to strides in the computation period, they provide a series of unrevealed states $h(t)$, as a representation of the past hidden state $h(t-1)$ and the input for position t . This essentially sequential nature prevents correlation within training examples, that turns into crucial at longer sequence lengths, as memory restraints limit clustering between examples. Novel work has accomplished important developments in computational effectiveness via factorization tricks and conditional computation, by further enhancing framework performance in the case of the latter. Nevertheless, the primary restriction of sequential computation exists.

Self-attention, often named intra-attention represents an attention mechanism linked to various positions of a unique sequence to compute a representation of the sequence. Self-attention has been implemented successfully in different tasks encompassing reading comprehension, abstractive summarization, textual entailment, and learning task-independent sentence representations. The recurrent attention mechanism is fundamental to an end-to-end memory network instead of sequence-aligned relapse and has been displayed to perform well on simple-language question answering and language modeling tasks.

3 METHODOLOGY

In this chapter we will show the methodology used to achieve the implementation of Arabic language in addition to ESPnet.

Figure 4 shows an overview of the project stages. To achieve the goal of adding a new language to ESPnet, first we should have the data through a dataset previously created. The next step is to process this dataset to be compatible with RNN for training. After preprocessing the metadata is extracted from the dataset. This metadata is fed for training in the next step. This training will result with a model used for TTS process. Finally, the model is integrated with a GUI to be used easily by unprofessional users. The GUI can also be integrated with pretrained models.

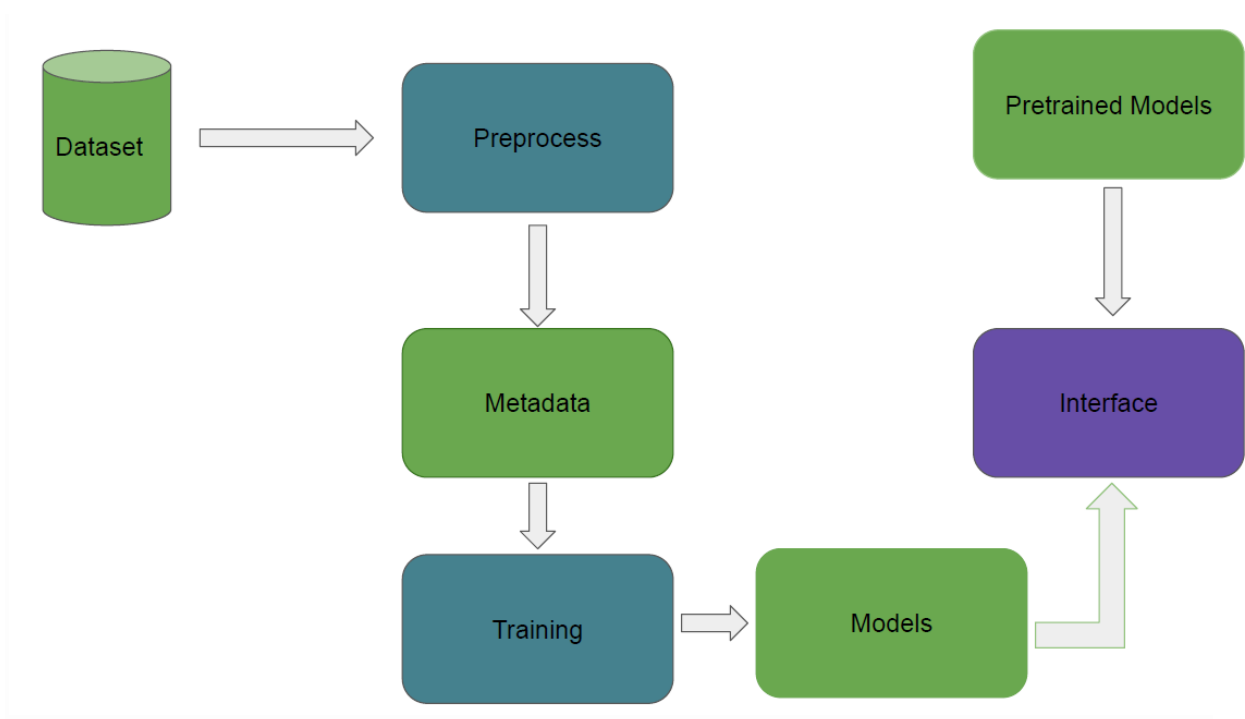


Figure 4 A schematic for the project

A pretrained model downloaded from the internet is used to test the software. A demo server that contains the GUI is used to interface with the user, as shown in figure 5.

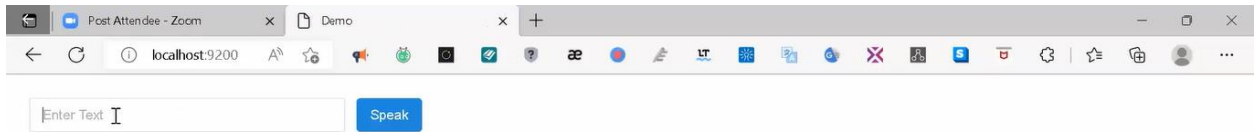


Figure 5 GUI used to synthesize the written text to speech

To activate the server, the command “python demo_server.py--checkpoint .\{folder_in_a_destination_of_your_choice}\model.ckpt-200000” must be issued from Anaconda first, and then a browser is opened and the local host with port 9200 is browsed to connect to the local server through “localhost:9200”. A virtual environment should be used for the system to work properly.

To create a virtual environment, we use Anaconda. Anaconda is a type of prompt that is similar to the classical computer command prompt (the cmd), as shown in figure 6.

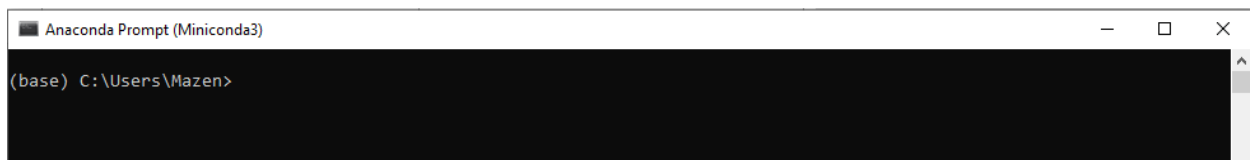


Figure 6 Anaconda prompt base virtual environment

The creation of a new environment is simple through the command “conda create -n {environment_name} python=3.5”, as shown in figure 7. Python version 3.5 is chosen for its compatibility with the needed Tensorflow library for the project. Finally, the environment is ready to be activated using the command “activate {environment_name}”, as shown in figure 8.

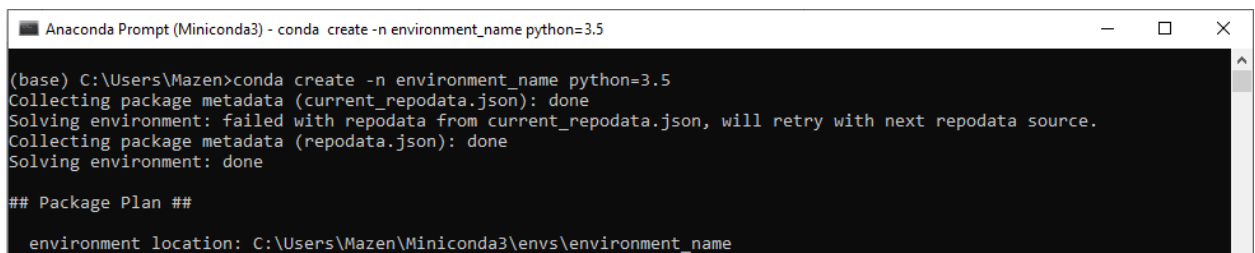
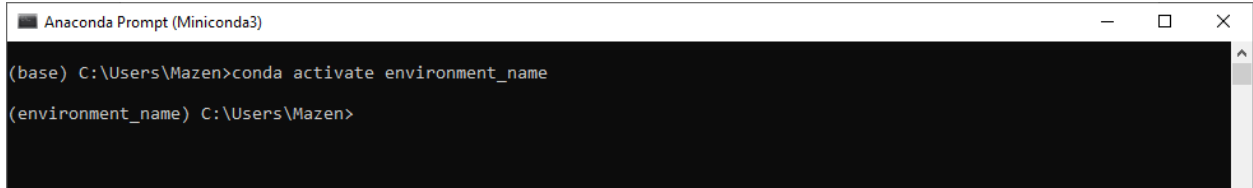


Figure 7 Creating a virtual environment in Anaconda



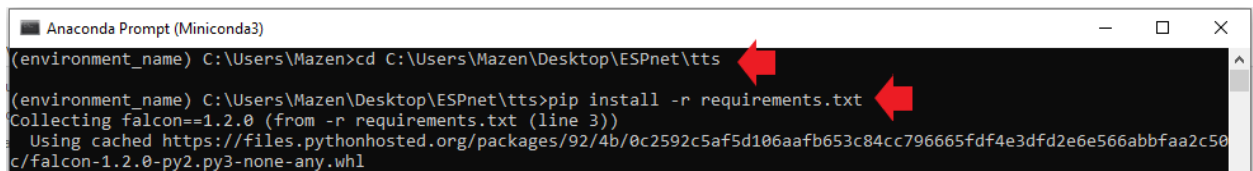
```

Anaconda Prompt (Miniconda3)
(base) C:\Users\Mazen>conda activate environment_name
(environment_name) C:\Users\Mazen>

```

Figure 8 Anaconda virtual environment activated

To have a fully functional system, some python libraries, including TensorFlow, should be downloaded. To download the needed libraries all at once, all libraries should be written to a text file (requirements.txt in our case) and the command “pip install -r requirements.txt” is then issued, as shown in figure 9.



```

Anaconda Prompt (Miniconda3)
(environment_name) C:\Users\Mazen>cd C:\Users\Mazen\Desktop\ESPnet\tts
(environment_name) C:\Users\Mazen\Desktop\ESPnet\tts>pip install -r requirements.txt
Collecting falcon==1.2.0 (from -r requirements.txt (line 3))
Using cached https://files.pythonhosted.org/packages/92/4b/0c2592c5af5d106aafb653c84cc796665fdf4e3dfd2e6e566abbfaa2c50c/falcon-1.2.0-py2.py3-none-any.whl

```

Figure 9 Python libraries installation in Anaconda

3.2 Steps to add a new language to the ESPnet TTS process.

3.2.1 Data preparation

A dataset was created for later training according to the needed language, Arabic. A Preprocessor is used to create a new dataset. A Preprocessor is a code that is responsible for generating an already spoken text, a mel-scale spectrogram of the audio, and a linear-scale spectrogram of the audio. These three components compose a dataset that is used for training.

For each training example, a preprocessor should:

- i. Load the audio file.
- ii. Compute linear-scale and mel-scale spectrograms (float32 numpy arrays).
- iii. Save the spectrograms to disk. Note that the transpose of the matrix returned by ``audio.Spectrogram`` is saved so that it's in time-major format.
- iv. Generate a tuple “(spectrogram filename, mel spectrogram filename, n_frames, text)” to write to train.txt. n_frames is just the length of the time axis of the spectrogram.


```

def _process_utterance(out_dir, index, wav_path, text):

    # Load the audio to a numpy array:
    wav = audio.load_wav(wav_path)

    # Compute the linear-scale spectrogram from the wav:
    spectrogram = audio.spectrogram(wav).astype(np.float32)
    n_frames = spectrogram.shape[1]

    # Compute a mel-scale spectrogram from the wav:
    mel_spectrogram = audio.melspectrogram(wav).astype(np.float32)

    # Write the spectrograms to disk:
    spectrogram_filename = 'nawar-spec-%05d.npy' % index
    mel_filename = 'nawar-mel-%05d.npy' % index
    np.save(os.path.join(out_dir, spectrogram_filename), spectrogram.T, allow_pickle=False)
    np.save(os.path.join(out_dir, mel_filename), mel_spectrogram.T, allow_pickle=False)

    # Return a tuple describing this training example:
    return (spectrogram_filename, mel_filename, n_frames, text)

```

Figure 0.10 Example of a Preprocessor function

3.2.2 Wav dump / Embedding preparation.

Wave files should be prepared to be used during training process. The Arabic corpus from Dr. Nawar AL HALABI Ph.D. is used. “librosa” is used by the Preprocessor to load wav files from our dataset to numpy array so mel-scale spectrogram can be computed and written to the disk.

After setting-up the Preprocessor and the needed files, the preprocessing can start using the command “python.\preprocess.py --dataset {dataset}”.

ARA NORM 0002.wav	1 479 972	1 313 695	2015-02-22 22:16	2016-10-14 13:18	2016-10-14 13:18	A	-	1440A7E3	Deflate
ARA NORM 0003.wav	638 768	583 439	2015-02-22 22:16	2016-10-14 13:18	2016-10-14 13:18	A	-	FBED953C	Deflate
ARA NORM 0004.wav	1 351 942	1 233 126	2015-02-22 22:16	2016-10-14 13:18	2016-10-14 13:18	A	-	1303FCF3	Deflate
ARA NORM 0005.wav	352 846	311 244	2015-02-22 22:16	2016-10-14 13:18	2016-10-14 13:18	A	-	D32FC17C	Deflate
ARA NORM 0006.wav	984 014	885 681	2015-02-22 22:16	2016-10-14 13:18	2016-10-14 13:18	A	-	9A0A4EB0	Deflate
ARA NORM 0007.wav	1 745 182	1 595 220	2015-02-22 22:16	2016-10-14 13:18	2016-10-14 13:18	A	-	38134E4A	Deflate
ARA NORM 0008.wav	1 746 882	1 582 532	2015-02-22 22:16	2016-10-14 13:18	2016-10-14 13:18	A	-	8F1AB8A0	Deflate
ARA NORM 0009.wav	1 604 298	1 455 438	2015-02-22 22:16	2016-10-14 13:18	2016-10-14 13:18	A	-	9C27C84A	Deflate
ARA NORM 0010.wav	458 638	407 895	2015-02-22 22:16	2016-10-14 13:18	2016-10-14 13:18	A	-	80FED803	Deflate
ARA NORM 0011.wav	418 926	384 009	2015-02-22 22:16	2016-10-14 13:18	2016-10-14 13:18	A	-	B109E846	Deflate
ARA NORM 0012.wav	358 522	318 261	2015-02-22 22:16	2016-10-14 13:18	2016-10-14 13:18	A	-	3F9D665A	Deflate
ARA NORM 0013.wav	738 862	669 708	2015-02-22 22:16	2016-10-14 13:18	2016-10-14 13:18	A	-	26D5ABE2	Deflate
ARA NORM 0014.wav	392 582	355 356	2015-02-22 22:16	2016-10-14 13:18	2016-10-14 13:18	A	-	F0D6A23C	Deflate
ARA NORM 0015.wav	866 330	792 177	2015-02-22 22:16	2016-10-14 13:18	2016-10-14 13:18	A	-	878F75AF	Deflate
ARA NORM 0016.wav	660 418	601 017	2015-02-22 22:16	2016-10-14 13:18	2016-10-14 13:18	A	-	CE44CE15	Deflate
ARA NORM 0017.wav	1 274 042	1 165 448	2015-02-22 22:16	2016-10-14 13:18	2016-10-14 13:18	A	-	B78B5CDF	Deflate
ARA NORM 0018.wav	1 642 746	1 485 034	2015-02-22 22:16	2016-10-14 13:18	2016-10-14 13:18	A	-	91DC9059	Deflate
ARA NORM 0019.wav	824 010	756 148	2015-02-22 22:16	2016-10-14 13:18	2016-10-14 13:18	A	-	D5FA8838	Deflate
ARA NORM 0020.wav	759 526	701 053	2015-02-22 22:16	2016-10-14 13:18	2016-10-14 13:18	A	-	900B1B38	Deflate
ARA NORM 0021.wav	1 100 106	1 000 175	2015-02-22 22:16	2016-10-14 13:18	2016-10-14 13:18	A	-	B232DA50	Deflate
ARA NORM 0022.wav	839 002	757 727	2015-02-22 22:16	2016-10-14 13:18	2016-10-14 13:18	A	-	89C0D477	Deflate
ARA NORM 0023.wav	1 332 344	1 224 440	2015-02-22 22:16	2016-10-14 13:18	2016-10-14 13:18	A	-	AE6FABA1	Deflate
ARA NORM 0024.wav	626 138	575 648	2015-02-22 22:16	2016-10-14 13:18	2016-10-14 13:18	A	-	1A0CB9C4	Deflate
ARA NORM 0025.wav	638 770	590 609	2015-02-22 22:16	2016-10-14 13:18	2016-10-14 13:18	A	-	E28D973D	Deflate
ARA NORM 0026.wav	1 460 342	1 306 513	2015-02-22 22:16	2016-10-14 13:18	2016-10-14 13:18	A	-	9E56E290	Deflate
ARA NORM 0027.wav	621 738	571 091	2015-02-22 22:16	2016-10-14 13:18	2016-10-14 13:18	A	-	6DA231F5	Deflate
ARA NORM 0028.wav	508 182	463 853	2015-02-22 22:16	2016-10-14 13:18	2016-10-14 13:18	A	-	273B13AE	Deflate
ARA NORM 0029.wav	1 584 170	1 437 266	2015-02-22 22:16	2016-10-14 13:18	2016-10-14 13:18	A	-	8EF8007B	Deflate
ARA NORM 0030.wav	1 833 430	1 647 731	2015-02-22 22:16	2016-10-14 13:18	2016-10-14 13:18	A	-	61982D6C	Deflate
ARA NORM 0031.wav	624 262	476 277	2015-04-01 16:45	2016-10-14 13:18	2016-10-14 13:18	A	-	E59CE30F	Deflate
ARA NORM 0032.wav	374 006	338 406	2015-02-22 22:16	2016-10-14 13:18	2016-10-14 13:18	A	-	8A5D98E7	Deflate

Figure 0.11 The Arabic corpus used for training.

3.2.3 TTS training

In this stage the RNN training of the ESPnet to understand Arabic language. By training the RNN the weights of the connections between each layer of the network are manipulated to have the lowest error between the actual and expected result.

The training starts after issuing the command “python.\train.py”. In this command we are running the train.py python code to train the RNN. Multiple attributes can be changed from default thanks to “argparse” library. The user can specify the working directory through “-base-dir” argument. Also, the user can specify training file directory through “-input” arguments. The model used can be specified through “--model”. The running name that is used for logging using “-name” argument. This name is used to differentiate between different models used in the logs. Hyperparameters can be modulated using the “--hparams”. These hyperparameter overrides as a comma-separated list of name=value pairs. To set the global step to restore from checkpoint the user can add the argument “--restore_step”. Additionally, the steps between running summary operations can be set by the argument “--summary_interval”, and the steps between writing checkpoints are set “--checkpoint_interval”. The user can use the help of webhook to have periodic reports about what’s happening in the background. To the Unified Resource Link (URL) for webhook “--slack_url” argument is used. The user can also specify the level of TensorFlow logs. This level means the amount of information that TensorFlow will prompt on the screen while training, and it can be set using the argument “--tf_log_level”.

```
def main():
    parser = argparse.ArgumentParser()
    parser.add_argument('--base_dir', default=os.path.expanduser('~/.tacotron'))
    parser.add_argument('--input', default='training/train.txt')
    parser.add_argument('--model', default='tacotron')
    parser.add_argument('--name', help='Name of the run. Used for logging. Defaults to model name.')
    parser.add_argument('--hparams', default='',
                        help='Hyperparameter overrides as a comma-separated list of name=value pairs')
    parser.add_argument('--restore_step', type=bool, default=True, help='Global step to restore from checkpoint.')
    parser.add_argument('--summary_interval', type=int, default=100,
                        help='Steps between running summary ops.')
    parser.add_argument('--checkpoint_interval', type=int, default=1000,
                        help='Steps between writing checkpoints.')
    parser.add_argument('--slack_url', help='Slack webhook URL to get periodic reports.')
    parser.add_argument('--tf_log_level', type=int, default=1, help='Tensorflow C++ log level.')
    parser.add_argument('--git', action='store_true', help='If set, verify that the client is clean.')
    args = parser.parse_args()
    os.environ['TF_CPP_MIN_LOG_LEVEL'] = str(args.tf_log_level)
    run_name = args.name or args.model
    log_dir = os.path.join(args.base_dir, 'logs-%s' % run_name)
    os.makedirs(log_dir, exist_ok=True)
    infolog.init(os.path.join(log_dir, 'train.log'), run_name, args.slack_url)
    hparams.parse(args.hparams)
    train(log_dir, args)

if __name__ == '__main__':
    main()
```

Figure 0.12 RNN training function

Figure 3.11 shows the architecture of the NN used in ESPnet. The network's first layer is a non-trainable layer whose function is only to take raw wave data as input and convert it to a mel-spectrogram. This spectrogram will be fed to an extraction layer to extract its features. This layer is made up of two consecutive 2D Convolutional layers (Conv2D). Conv2D itself is made up of three layer types as follows: input, convolutional, and pooling layers (Agiomyrgiannakis 2015). CNN is mainly used for image classification [79], but by adding the attention mechanism the limitations of using CNN in text classification can be addressed [80]. An attention function is made up of 4 vectors: a query, a keys vector, a values vector, and an output. It's a function where the mapping of the query and a set of key-value pairs to the output occurs. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key After features extraction (Ruzsics & Samardžić 2019).

The resulting data from the Conv2D layer then goes to a Long-term Dependency layer. This layer gives the NN the ability to memorize for an interval of time. This memory increases the efficiency of the NN. The resulting data from this layer continues to an Attention layer that uses the Softmax activation function. Softmax is efficient when dealing with multidimensional spaces and multi-hidden layers. It restricts the probability between 0 and 1 only (H. Zhang et al. 2019). Finally, the data goes to the Classification layer. In this layer the classification for the voice signal from the raw wav file takes place to know what the letters is spoken.

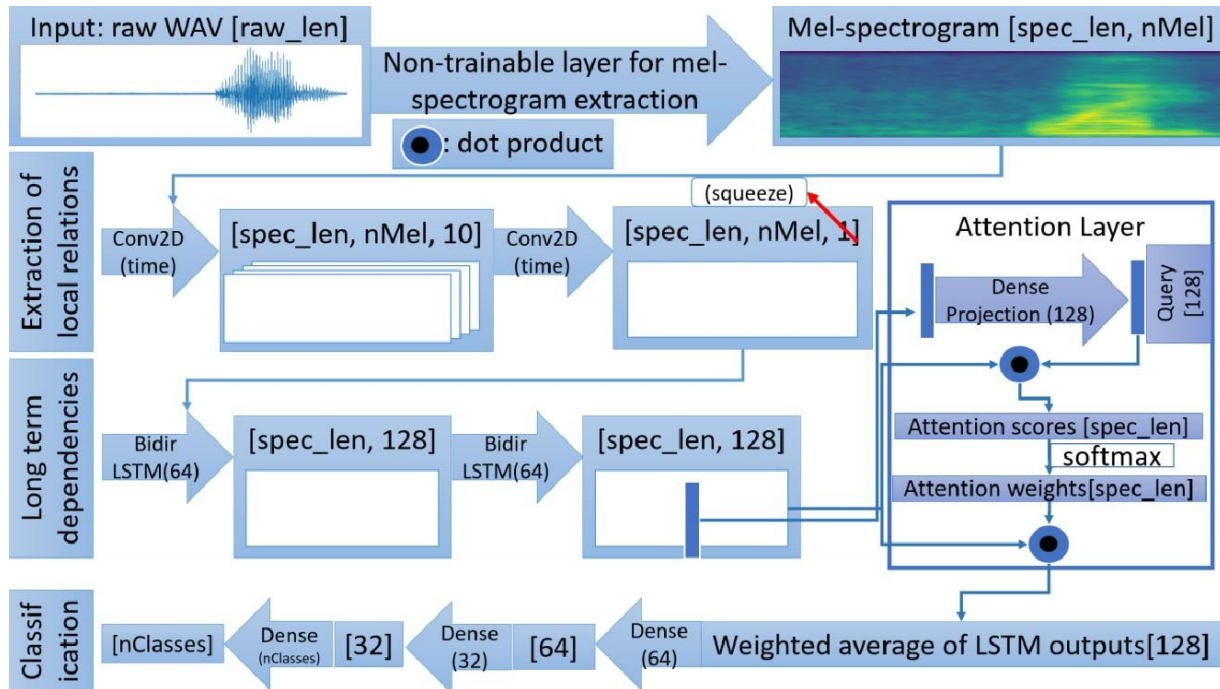


Figure 0.13 ESPnet RNN architecture

3.2.4 TTS synthesizing

Figure 10 shows an overview of the process of speech synthesis. The process starts with text normalization. Its job is to resolve issues such as numbers, abbreviations, and currency symbols. Normalizing a text means to convert it to its standard form.

The second step is Phonetization. It is made up of two stages. The first stage is converting input text into phonemes. The second stage is converting the phonemes into phones. This means converting each letter to its sound (Letter-To-Sound LTS).

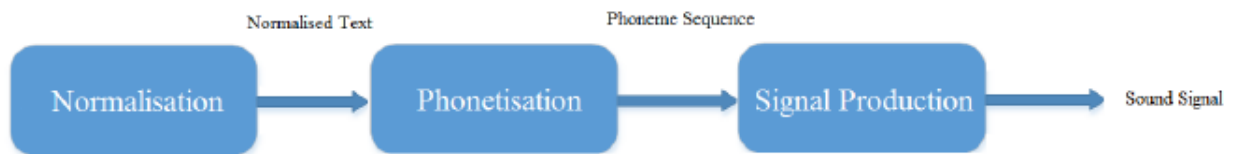


Figure 0.14 Overview of process of speech synthesis

3.2.5 Link to the server

The GUI (figure 5) used is a demo server with a text field to write the sentence needed to be synthesized and a Push button to start the algorithm for synthetization. The code based on HTML is used to connect the core algorithms of the ESPnet to the GUI server.

```
html_body = '''<html><title>Demo</title>
<style>
body {padding: 16px; font-family: sans-serif; font-size: 14px; color: #444}
input {font-size: 14px; padding: 8px 12px; outline: none; border: 1px solid #ddd}
input:focus {box-shadow: 0 1px 2px rgba(0,0,0,.15)}
p {padding: 12px}
button {background: #28d; padding: 9px 14px; margin-left: 8px; border: none; outline: none;
        color: #fff; font-size: 14px; border-radius: 4px; cursor: pointer;}
button:hover {box-shadow: 0 1px 2px rgba(0,0,0,.15); opacity: 0.9;}
button:active {background: #29f;}
button[disabled] {opacity: 0.4; cursor: default}
</style>
<body>
<form>
  <input id="text" type="text" size="40" placeholder="Enter Text">
  <button id="button" name="synthesize">Speak</button>
</form>
<p id="message"></p>
<audio id="audio" controls autoplay hidden></audio>
<script>
function q(selector) {return document.querySelector(selector)}
q('#text').focus()
q('#button').addEventListener('click', function(e) {
  text = q('#text').value.trim()
  if (text) {
    q('#message').textContent = 'Synthesizing...'
    q('#button').disabled = true
    q('#audio').hidden = true
    synthesize(text)
  }
  e.preventDefault()
  return false
})
function synthesize(text) {
  fetch('/synthesize?text=' + encodeURIComponent(text), {cache: 'no-cache'})
    .then(function(res) {
      if (!res.ok) throw Error(res.statusText)
      return res.blob()
    }).then(function(blob) {
      q('#message').textContent = ''
      q('#button').disabled = false
      q('#audio').src = URL.createObjectURL(blob)
      q('#audio').hidden = false
    }).catch(function(err) {
```

Figure 0.15 HTML-based server function

```

class UIResource:
    def on_get(self, req, res):
        res.content_type = 'text/html'
        res.body = html_body

class SynthesisResource:
    def on_get(self, req, res):
        if not req.params.get('text'):
            raise falcon.HTTPBadRequest()
        res.data = synthesizer.synthesize(req.params.get('text'))
        res.content_type = 'audio/wav'

synthesizer = Synthesizer()
api = falcon.API()
api.add_route('/synthesize', SynthesisResource())
api.add_route('/', UIResource())

if __name__ == '__main__':
    from wsgiref import simple_server
    parser = argparse.ArgumentParser()
    parser.add_argument('--checkpoint', required=True, help='Full path to model checkpoint')
    parser.add_argument('--port', type=int, default=9200)
    parser.add_argument('--hparams', default='',
        help='Hyperparameter overrides as a comma-separated list of name=value pairs')
    args = parser.parse_args()
    os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
    hparams.parse(args.hparams)
    print(hparams.debug_string())
    synthesizer.load(args.checkpoint)
    print('Serving on port %d' % args.port)
    simple_server.make_server('0.0.0.0', args.port, api).serve_forever()
else:
    synthesizer.load(os.environ['CHECKPOINT'])

```

Figure 0.16 Code to connect to the GUI server.

3.3 IMPORTANT LIBRARIES TO INSTALL

Arabic_pronounce library (Figure 3.15) is one of the important libraries that are needed to be downloaded for the system to work properly. This library is used to phonetize the Arabic words, as shown in Figure 3.16. According to Dr. Nawar, the library developer, official github, the library converts Arabic diacritice text to a sequence of phonemes and create a pronunciation dictionary from them for alignment.

3.4 SYSTEM LAUNCH

To launch the system, the command “python entry-point.py --checkpoint .\tm\model.chkpt-200000” should be issued in the virtual environment. The command is made up of four parts. First the “python” specifies the programming language of code to be launched. The “entry-point.py” argument is the name of the main python file to be launched. This file will run other branch files according to its need. The third argument specifies the option that we want to start from a checkpoint. This checkpoint is created after a specific number of iterations of training occurred. The number of iterations can be known from the name of the checkpoint itself. The argument “.tm\model.chkpt-200000” shows the directory of the checkpoint file. At the end of the checkpoint, we can read the number of iterations on this point, which is 200000 iterations.

4. FINDINGS

The demographic characteristics of the respondents are presented in Table 4.1. Thirty-six (52.9%) of the participants are males whereas there were 32 (47.1%) females. Most respondents (33.8%) are between 26 and 35 years old, 18% of the respondents are between 36 and 45 years of age, and 16 % are between 46 and 55 years. Most respondents (48.5%) have a bachelor’s degree, while 23.1% have a master’s degree and 14.7% have a Ph.D. degree. The results also indicated that 43 (89.7%) have Arabic as their native language.

The findings of the measurement model are presented in Table 4.2. To examine the internal consistency of each variable, the Cronbach alpha was evaluated, thus confirming the reliability of the measurement items. The reliability of the constructs assessing the quality and acceptance of the TTS and the human voice is 0.79 and 0.7, respectively as illustrated in Table 4.2, thus confirming internal consistency. The findings demonstrate the internal consistency of the 9-item scale.

To assess validity in this study, all item loadings (λ), and correlations of the scale on the overall quality and acceptance of the TTS and human voice are assessed and presented in Table 4.2. To confirm convergent validity, all item loadings (λ) are more than the threshold of 0.6 (Fornell & Larcker, 1981). The nine-item scale demonstrates a high correlation with acceptance (0.55, $p < 0.01$) and overall quality

(0.65, $p < 0.01$) for the TTS and a high correlation with the single item on acceptance (0.65, $p < 0.01$) and overall quality (0.81, $p < 0.01$) for the human voice as shown in Table 4.2

Table 4.1 Respondents' demographic profile (N= 68)

Demographic information	Frequency	Percentage
Gender		
Male	36	52.9%
Female	32	47.1%
Age		
Under 16	1	1.5%
16- 25	7	10.3%
26 - 35	23	33.8%
36 - 45	18	26.5%
46 -55	16	23.5%
Above 56	3	4.4%
Educational status		
Bachelor's degree	33	48.5%
High school	6	8.8%
Master's degree	15	23.1%
Ph.D. degree	10	14.7%
Vocational degree	4	5.9%
Mother language		
Arabic language	61	89.7%
English language	1	1.5%
Another language	6	8.8%

Table 4.1 Measurement Model

Items	Assessing TTS Quality				Assessing Human voice			
	Loading s(λ)	Mea n	Standar d deviatio n	Item to total correlat ion	Loading s(λ)	Mea n	Standar d deviatio n	Item to total correlation
Listening effort	0.742	2.94	1.423	0.75	0.833	2.18	0.961	0.78
Pronunciation	0.612	2.84	1.410	0.61	0.651	1.74	1.277	0.69
Speaking rate	0.602	3.28	1.144	0.79	0.845	2.06	0.751	0.88
Pleasantness	0.720	4.41	2.307	0.69	0.728	2.01	1.311	0.76
naturalness	0.744	3.62	1.305	0.81	0.731	2.34	1.462	0.69
Audio flow	0.658	2.29	1.210	0.90	0.746	1.44	0.999	0.82
Ease of listening	0.615	3.40	2.317	0.73	0.713	1.46	0.961	0.68
Comprehension problems	0.607	3.25	1.309	0.65	0.602	1.65	0.967	0.77
Articulation	0.615	2.41	1.175	0.78	0.727	2.11	1.152	0.79

The results presented in the tables below were generated using SPSS program and were interpreted utilizing descriptive statistical methods such as the frequency, percent, valid percent, and cumulative percent. Frequency represents the number of occurrences of each response chosen by the participants. Furthermore, the percent is a relative value indicating hundredth parts of any response. Moreover, the valid percent refers to the percentage after excluding missing cases. It is essential to highlight that the percent and valid percent values were identical since there are no missing values. Over and above, the cumulative percent represents the percentage of the cumulative frequency within each interval.

The overall quality and acceptance of the TTS audio were compared to that of the human voice in Tables 4.3 and 4.4, respectively. As presented in Table 4.3, 48.6% of the respondents reported that the overall quality of the TTS voice is either poor or very poor; however, 14.7% reported that the quality of the TTS

is excellent. On the other hand, approximately 65% and 24% of the respondents claimed that the overall quality of the human voice was excellent and good respectively; meanwhile, approximately 6% reported that the quality of the human voice is very poor.

Table 4.2 Quality rating for TTS vs human voice

(Quality of the TTS)					
Item Scale		Frequency	Percent	Valid Percent	Cumulative Percent
Excellent		10	14.7	14.7	14.7
Fair		13	19.1	19.1	33.8
Good		12	17.6	17.6	51.5
Poor		18	26.5	26.5	77.9
Very Poor		15	22.1	22.1	100.0
Total		68	100.0	100.0	

(Quality of the human voice)					
Item Scale		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Excellent	44	64.7	64.7	64.7
	Fair	1	1.5	1.5	66.2
	Good	16	23.5	23.5	89.7
	Poor	3	4.4	4.4	94.1
	Very Poor	4	5.9	5.9	100.0
Total		68	100.0	100.0	

Concerning the acceptance of audios, respondents determined if the audios could be utilized for interactive telephone or wireless service systems and their responses are shown in Table 4.4. Specifically, approximately 62% of the respondents reported negative responses concerning the TTS audio. Meanwhile, 88.2% of the respondents agreed to utilize the human voice for information service systems.

Table 4.3 Acceptance of audio (TTS audio versus human audio)
(Acceptance of the TTS)

Item scale	Frequency	Percent	Valid Percent	Cumulative Percent
No	42	61.8	61.8	61.8
Yes	26	38.2	38.2	100.0
Total	68	100.0	100.0	

(Acceptance of the human voice)

Item scale	Frequency	Percent	Valid Percent	Cumulative Percent
No	8	11.8	11.8	11.8
Yes	60	88.2	88.2	100.0
Total	68	100.0	100.0	

As presented in Table 4.5, approximately, 30% of the respondents claimed that moderate effort is required to understand the message in the TTS audio. Furthermore, 25% of the participants argued that attention is necessary however no appreciable effort is required to understand the TTS message. On the other hand, approximately 65% of the respondents reported that no effort is required with complete relaxation to understand the human voice message.

Table 4.4 Listening effort (TTS audio versus human voice)
(Concerning the TTS audio)

Item scale	Frequency	Percent	Valid Percent	Cumulative Percent
Attention necessary; no appreciable effort required	17	25.0	25.0	25.0
Complete relaxation possible; no effort required	9	13.2	13.2	38.2
Considerable effort required	13	19.1	19.1	57.4
Moderate effort required	19	27.9	27.9	85.3
No meaning understood with any feasible effort	10	14.7	14.7	100.0
Total	68	100.0	100.0	

(Concerning the human voice audio)

Item scale	Frequency	Percent	Valid Percent	Cumulative Percent
Attention necessary; no appreciable effort required	12	17.6	17.6	17.6
Complete relaxation possible; no effort required	44	64.7	64.7	82.4
Considerable effort required	2	2.9	2.9	85.3
Moderate effort required	8	11.8	11.8	97.1
No meaning understood with any feasible effort	2	2.9	2.9	100.0
Total	68	100.0	100.0	

Table 4.6 demonstrates the responses regarding anomalies in pronunciation. Twenty-nine percent of the respondents noticed slightly annoying anomalies in pronunciation while listening to the TTS audio; however, 16.2% of the respondents noticed annoying anomalies in pronunciation.

On the other hand, 72.1% percent of the respondents did not notice anomalies in pronunciation while they were listening to the human voice.

Table 4.5 Pronunciation (TTS audio versus human voice)
(Regarding the TTS audio)

Item scale	Frequency	Percent	Valid Percent	Cumulative Percent
No	18	26.5	26.5	26.5
Yes, annoying	11	16.2	16.2	42.6
Yes, but not annoying	11	16.2	16.2	58.8
Yes, slightly annoying	20	29.4	29.4	88.2
Yes, very annoying	8	11.8	11.8	100.0
Total	68	100.0	100.0	

(Regarding the human voice)

Item scale	Frequency	Percent	Valid Percent	Cumulative Percent
No	49	72.1	72.1	72.1
Yes, annoying	2	2.9	2.9	75.0
Yes, but not annoying	6	8.8	8.8	83.8
Yes, slightly annoying	8	11.8	11.8	95.6
Yes, very annoying	3	4.4	4.4	100.0
Total	68	100.0	100.0	

Over and above, respondents have evaluated the average speed of delivery while listening to the TTS audio and human voice. For instance, 29.4% of the respondents claimed that the average speed of delivery was just right while listening to the TTS audio; whereas, approximately 28% stated that the average rate of delivery was slightly fast or slightly slow as shown in Table 4.7.

On the other hand, approximately 62% of the respondents reported that the average speed of delivery was just right while they listened to the human voice, while 19.1% claimed that the average speed was very fast or very slow.

Table 4.6 Speaking rate (TTS audio versus human voice)

The average speed of delivery was: (For the TTS)

Item scale	Frequency	Percent	Valid Percent	Cumulative Percent
Extremely Fast or Extremely Slow	4	5.9	5.9	5.9
Fairly Fast or Fairly Slow	14	20.6	20.6	26.5
Just Right	20	29.4	29.4	55.9
Slightly Fast or Slightly Slow	19	27.9	27.9	83.8
Very Fast or Very Slow	11	16.2	16.2	100.0
Total	68	100.0	100.0	

The average speed of delivery was: (For the human voice)

Item scale	Frequency	Percent	Valid Percent	Cumulative Percent
Extremely Fast or Extremely Slow	2	2.9	2.9	2.9
Fairly Fast or Fairly Slow	9	13.2	13.2	16.2
Just Right	42	61.8	61.8	77.9
Slightly Fast or Slightly Slow	13	19.1	19.1	97.1
Very Fast or Very Slow	2	2.9	2.9	100.0
Total	68	100.0	100.0	

Concerning pleasantness, the results presented in Table 4.8 revealed that 33.8% of the respondents claimed that the TTS audio was unpleasant. However, 42.6% of the respondents claimed that the human audio was very pleasant and approximately 28% argued that it is pleasant.

Table 4.7 Pleasantness (TTS audio versus human voice)
(Regarding the TTS audio)

Item scale	Frequency	Percent	Valid Percent	Cumulative Percent
Neutral	25	36.8	36.8	36.8
Pleasant	8	11.8	11.8	48.5
Unpleasant	23	33.8	33.8	82.4
Very Pleasant	6	8.8	8.8	91.2
Very Unpleasant	6	8.8	8.8	100.0
Total	68	100.0	100.0	

(Regarding the human voice)

Item scale	Frequency	Percent	Valid Percent	Cumulative Percent
Neutral	16	23.5	23.5	23.5
Pleasant	19	27.9	27.9	51.5
Unpleasant	2	2.9	2.9	54.4
Very Pleasant	29	42.6	42.6	97.1
Very Unpleasant	2	2.9	2.9	100.0
Total	68	100.0	100.0	

As presented in Table 4.9, 32.4% of the respondents reported that the TTS audio was unnatural and approximately 12% reported that it was very unnatural. On the other hand, 35.5% of the respondents claimed that the human voice audio was very natural and 50% claimed that it was natural.

Table 4.8 Naturalness (TTS audio versus human voice)
(Concerning the TTS audio)

Item scale	Frequency	Percent	Valid Percent	Cumulative Percent
Natural	18	26.5	26.5	26.5
Neutral	13	19.1	19.1	45.6
Unnatural	22	32.4	32.4	77.9
Very Natural	7	10.3	10.3	88.2
Very Unnatural	8	11.8	11.8	100.0
Total	68	100.0	100.0	

(Concerning the human voice)

Item scale	Frequency	Percent	Valid Percent	Cumulative Percent
Natural	34	50.0	50.0	50.0
Neutral	5	7.4	7.4	57.4
Unnatural	3	4.4	4.4	61.8
Very Natural	24	35.3	35.3	97.1
Very Unnatural	2	2.9	2.9	100.0
Total	68	100.0	100.0	

Based on the results presented in Table 4.10, 35.3 % of the participants reported that the flow of the TTS audio was either discontinuous or very discontinuous. Meanwhile, approximately 81% of the participants reported that the human voice audio was either smooth or very smooth.

Table 4.9 Audio flow (TTS audio versus human voice)
(Concerning the TTS audio)

Item scale	Frequency	Percent	Valid Percent	Cumulative Percent
Discontinuous	20	29.4	29.4	29.4
Neutral	24	35.3	35.3	64.7
Smooth	14	20.6	20.6	85.3
Very Discontinuous	4	5.9	5.9	91.2
Very Smooth	6	8.8	8.8	100.0
Total	68	100.0	100.0	

(Concerning the human voice)

Item scale	Frequency	Percent	Valid Percent	Cumulative Percent
Discontinuous	3	4.4	4.4	4.4
Neutral	8	11.8	11.8	16.2
Smooth	34	50.0	50.0	66.2
Very Discontinuous	2	2.9	2.9	69.1
Very Smooth	21	30.9	30.9	100.0
Total	68	100.0	100.0	

The results presented in Table 4.11 revealed that 33.8% of the participants claimed that it was difficult to listen to the TTS audio for long periods and 13.2 % claimed that it was very difficult. On the other hand, approximately 47% and 31% of the respondents claimed that it was either very easy or easy to listen to the human voice for a long period.

Table 4.10 Ease of listening (TSS audio versus human voice)
(Concerning the TTS audio)

Item scale	Frequency	Percent	Valid Percent	Cumulative Percent
Difficult	23	33.8	33.8	33.8
Easy	16	23.5	23.5	57.4
Neutral	14	20.6	20.6	77.9
Very Difficult	9	13.2	13.2	91.2
Very Easy	6	8.8	8.8	100.0
Total	68	100.0	100.0	

(Concerning the human voice)

Item scale	Frequency	Percent	Valid Percent	Cumulative Percent
Difficult	7	10.3	10.3	10.3
Easy	21	30.9	30.9	41.2
Neutral	6	8.8	8.8	50.0
Very Difficult	2	2.9	2.9	52.9
Very Easy	32	47.1	47.1	100.0
Total	68	100.0	100.0	

Table 4.12 shows that 51.5 % of the respondents find, often and occasionally, certain words that were hard to be understood while listening to the TTS audio. However, 76.4% of the respondents never or rarely find hard words when they listened to the human voice.

Table 4.11 Comprehension problems (TTS audio versus human voice)
(In the TTS audio)

Item scale	Frequency	Percent	Valid Percent	Cumulative Percent
All the time	9	13.2	13.2	13.2
Never	10	14.7	14.7	27.9
Occasionally	18	26.5	26.5	54.4
Often	17	25.0	25.0	79.4
Rarely	14	20.6	20.6	100.0
Total	68	100.0	100.0	

(In the human voice)

Item scale	Frequency	Percent	Valid Percent	Cumulative Percent
All the time	3	4.4	4.4	4.4
Never	43	63.2	63.2	67.6
Occasionally	6	8.8	8.8	76.5
Often	7	10.3	10.3	86.8
Rarely	9	13.2	13.2	100.0
Total	68	100.0	100.0	

Concerning the articulation, Table 4.13 revealed that approximately 40% of the respondents reported that the TTS audio was less clear and 13.2 % reported that it was much less clear. Meanwhile, 47.1% of the respondents found the human voice very clear and approximately 30% indicated that it was clear.

Table 4.12 Articulation (TTS versus human voice)
Were the sounds in the TTS audio distinguishable?

Item scale	Frequency	Percent	Valid Percent	Cumulative Percent
Clear	16	23.5	23.5	23.5
Less Clear	27	39.7	39.7	63.2
Much Less Clear	9	13.2	13.2	76.5
Neutral	13	19.1	19.1	95.6
Very Clear	3	4.4	4.4	100.0
Total	68	100.0	100.0	

Were the sounds in the human voice audio distinguishable?

Item scale	Frequency	Percent	Valid Percent	Cumulative Percent
Clear	19	27.9	27.9	27.9
Less Clear	6	8.8	8.8	36.8
Much Less Clear	2	2.9	2.9	39.7
Neutral	9	13.2	13.2	52.9
Very Clear	32	47.1	47.1	100.0
Total	68	100.0	100.0	

4.3 DISCUSSION

Relying on the data analysis presented in Section 4.2.1, the findings revealed that the human voice overall quality is better than that of the TTS voice. Over and above, the results indicated that the human voice is better utilized for interactive telephone or wireless systems in comparison to the TTS audio. Moreover, the human audio seemed to exhibit better results than the TTS audio concerning all measurement items.

As mentioned earlier, end-to-end speech processing toolkit was utilized primarily in speech processing sections through open-source applications that encourage both TTS and ASR (Kwon et al. 2019; Tran 2020). Furthermore, in the new open-source End-to-end speech processing toolkit, the ESPnet, intends to

offer an endless neural end-to-end system for speech proceeding. However, the Arabic language, unlike other languages, is still not supported the way it should be in today's technology and application. There are a few applications which support TTS for Arabic language. The problem is that these applications are not easily extendable. To be able to add a new language such as Arabic, the API or used platform must be dynamically trained. ESPnet is one of the platforms that use Deep Learning (DL) for TTS and various other applications. This makes it easier to add the Arabic language to ESPnet, thus to the applications that use it for TTS.

Based on the papers' findings ,22.6% respondents stated that the quality of the TTS that they heard is poor, however, 14.7% and 17.6% participants claimed that the TTS was excellent and good, respectively. Moreover, although 61.8% of the respondents claimed that this TTS voice could not be used for an interactive service system, 38.2 % of the participants accepted the implementation of such TTS voice. Such findings revealed that the TTS voice overall quality and acceptance range would be perceived either good or excellent by a portion of the sample being targeted. This finding could be related to the fact that the Arabic language accent would differ between different Arabic countries. Accordingly, this report would suggest further implications to advance the overall quality and acceptance of TTS audio by utilizing the formal and standard Arabic language.

5. CONCLUSION

This chapter represents the methodology being adopted to compare a sample of synthetic speech (the Arabic TTS ESPnet) with a natural speech sample via the mean opinion scale (MOS) (Viswanathan & Viswanathan 2004). The data collection process was done utilizing a web-based survey on the Google Form platform. The survey was divided into four sections as follows: a section which introduces the purpose of the study and encouraged the voluntary participation of respondents, a section including questions related to the respondents' profiles, a section containing measurement items that assess the text-to-speech quality in the Arabic language, and a section assessing the quality of the human voice, also, in the Arabic language. The survey was distributed to respondents via WhatsApp application. The Google Form link was sent to participants aged between 15 and 60 years old. Moreover, data was collected from participants whose mother language is Arabic or who can understand the Arabic language. Each respondent listened to a synthesized sentence from the system in Section 3 and a natural speech sample in Section 4 and was guided to fill out the questionnaire. Collected data were analyzed via SPSS. The

number of respondents who participated in the study was 68. The findings revealed that the human voice over quality is better than that of the TTS voice and that the human voice is better in interactive service systems in comparison to the TTS audio. Meanwhile, the results indicated that, as well, the TTS voice overall quality and acceptance range would be perceived either good or excellent by a portion of the sample being targeted. Finally, this report would suggest further implications to advance the overall quality and acceptance of TTS audio by utilizing the formal and standard Arabic language.

REFERENCES

- Agiomyrgiannakis, Y. 2015. Vocaine the vocoder and applications in speech synthesis. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. pp. 4230–4234. IEEE.
- Arik, S.Ö., Chrzanowski, M., Coates, A., Diamos, G., Gibiansky, A., Kang, Y., Li, X., Miller, J., Ng, A. & Raiman, J. 2017. Deep voice: Real-time neural text-to-speech. In *International Conference on Machine Learning*. pp. 195–204. PMLR.
- Bahdanau, D., Cho, K. & Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. *ArXiv Preprint ArXiv:1409.0473*.
- Bigi, B. 2014. A Multilingual Text Normalization Approach. Human Language Technology Challenges for Computer Science and Linguistics, 8387. Springer Berlin Heidelberg.
- Bisani, M. & Ney, H. 2008. Joint-sequence models for grapheme-to-phoneme conversion. *Speech Communication* 50(5): 434–451.
- Black, A.W., Lenzo, K. & Pagel, V. 1998. Issues in building general letter to sound rules. In *The Third ESCA/COCOSDA Workshop (ETRW) on Speech Synthesis*.
- Bluche, T., Ney, H. & Kermorvant, C. 2013. Tandem HMM with convolutional neural network for handwritten word recognition. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. pp. 2390–2394. IEEE.
- Cer, D., Yang, Y., Kong, S., Hua, N., Limtiaco, N., John, R.S., Constant, N., Guajardo-Cespedes, M., Yuan, S. & Tar, C. 2018. Universal sentence encoder. *ArXiv Preprint ArXiv:1803.11175*.
- Chae, M.-J., Park, K., Bang, J., Suh, S., Park, J., Kim, N. & Park, L. 2018. Convolutional sequence to sequence model with non-sequential greedy decoding for grapheme to phoneme conversion. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. pp. 2486–2490. IEEE.
- Chan, W., Jaitly, N., Le, Q. & Vinyals, O. 2016. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. pp. 4960–4964. IEEE.
- Cheng, J., Dong, L. & Lapata, M. 2016. Long short-term memory-networks for machine reading. *ArXiv Preprint ArXiv:1601.06733*.
- Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H. & Bengio, Y. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *ArXiv Preprint ArXiv:1406.1078*.

- Chung, J., Gulcehre, C., Cho, K. & Bengio, Y. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *ArXiv Preprint ArXiv:1412.3555*.
- Conneau, A., Schwenk, H., Barrault, L. & Lecun, Y. 2016. Very deep convolutional networks for text classification. *ArXiv Preprint ArXiv:1606.01781*.
- Donahue, J., Dieleman, S., Bińkowski, M., Elsen, E. & Simonyan, K. 2020. End-to-end adversarial text-to-speech. *ArXiv Preprint ArXiv:2006.03575*.
- Dong, L., Wei, F., Zhou, M. & Xu, K. 2015. Question answering over freebase with multi-column convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. pp. 260–269.
- Elovitz, H., Johnson, R., McHugh, A. & Shore, J. 1976a. Letter-to-sound rules for automatic translation of english text to phonetics. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 24(6): 446–459. <http://ieeexplore.ieee.org/document/1162873/>.
- Elovitz, H., Johnson, R., McHugh, A. & Shore, J. 1976b. to-sound rules for automatic translation of English text to phonetics. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 24(6): 446–459.
- Galescu, L. & Allen, J.F. 2002. Pronunciation of proper names with a joint n-gram model for bi-directional grapheme-to-phoneme conversion. In *Seventh International Conference on Spoken Language Processing*.
- Gehring, J., Auli, M., Grangier, D. & Dauphin, Y.N. 2016. A convolutional encoder model for neural machine translation. *ArXiv Preprint ArXiv:1611.02344*.
- Gehring, J., Auli, M., Grangier, D., Yarats, D. & Dauphin, Y.N. 2017. Convolutional sequence to sequence learning. In *International Conference on Machine Learning*. pp. 1243–1252. PMLR.
- Gokcen, A., Zhang, H. & Sproat, R. 2019. Dual Encoder Classifier Models as Constraints in Neural Text Normalization. In *INTERSPEECH*. pp. 4489–4493.
- 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*. pp. 369–376.
- Graves, A., Jaitly, N. & Mohamed, A. 2013. Hybrid speech recognition with deep bidirectional LSTM. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*. pp. 273–278. IEEE.
- Greff, K., Srivastava, R.K., Koutník, J., Steunebrink, B.R. & Schmidhuber, J. 2016. LSTM: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems* 28(10): 2222–2232.
- Gregor, K., Danihelka, I., Graves, A., Rezende, D. & Wierstra, D. 2015. Draw: A recurrent neural network for image generation. In *International Conference on Machine Learning*. pp. 1462–1471. PMLR.

- Gupta, P., Schütze, H. & Andrassy, B. 2016. Table filling multi-task recurrent neural network for joint entity and relation extraction. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. pp. 2537–2547.
- Hassanien, A. E., Tolba, M., & Azar, A.T. 2021. *Advanced Machine Learning Technologies and Applications*. Springer International Publishing.
- Hayashi, T., Yamamoto, R., Inoue, K., Yoshimura, T., Watanabe, S., Toda, T., Takeda, K., Zhang, Y. & Tan, X. 2020. Espnet-TTS: Unified, Reproducible, and Integratable Open Source End-to-End Text-to-Speech Toolkit. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. pp. 7654–7658. IEEE <https://ieeexplore.ieee.org/document/9053512/>.
- He, K., Zhang, X., Ren, S. & Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pp. 770–778.
- Hochreiter, S. & Schmidhuber, J. 1997. Long short-term memory. *Neural Computation* 9(8): 1735–1780.
- Ioffe, S. & Szegedy, C. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*. pp. 448–456. PMLR.
- Juzová, M., Tihelka, D. & Vít, J. 2019. Unified Language-Independent DNN-Based G2P Converter. In *INTERSPEECH*. pp. 2085–2089.
- Krizhevsky, A., Sutskever, I. & Hinton, G.E. 2017. Imagenet classification with deep convolutional neural networks. *Communications of the ACM* 60(6): 84–90.
- Kuchaiev, O. & Ginsburg, B. 2017. Factorization tricks for LSTM networks. *ArXiv Preprint ArXiv:1703.10722*.
- Kwon, O., Jang, I., Ahn, C. & Kang, H.-G. 2019. Emotional Speech Synthesis Based on Style Embedded Tacotron2 Framework. In *2019 34th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC)*. pp. 1–4. IEEE <https://ieeexplore.ieee.org/document/8793393/>.
- LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11): 2278–2324.
- Lee, D.D., Pham, P., Largman, Y. & Ng, A. 2009. Advances in neural information processing systems 22. Report No. . Tech. Rep., Tech. Rep.
- Lee, J., Cho, K. & Hofmann, T. 2017. Fully character-level neural machine translation without explicit segmentation. *Transactions of the Association for Computational Linguistics* 5: 365–378.
- Lehnen, P., Allauzen, A., Lavergne, T., Yvon, F., Hahn, S. & Ney, H. 2013. Structure learning in hidden conditional random fields for grapheme-to-phoneme conversion. In *Annual Conference of the International Speech Communication Association*.

- Lehnen, P., Hahn, S., Guta, V.-A. & Ney, H. 2012. Hidden conditional random fields with m-to-n alignments for grapheme-to-phoneme conversion. In *Thirteenth Annual Conference of the International Speech Communication Association*.
- Li, X., Chen, Y.-N., Li, L., Gao, J. & Celikyilmaz, A. 2017. End-to-end task-completion neural dialogue systems. *ArXiv Preprint ArXiv:1703.01008*.
- Lu, L., Zhang, X. & Renais, S. 2016. On training the recurrent neural network encoder-decoder for large vocabulary end-to-end speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. pp. 5060–5064. IEEE.
- Lu, Y., Keung, P., Zhang, S., Sun, J. & Bhardwaj, V. 2017. A practical approach to dialogue response generation in closed domains. *ArXiv Preprint ArXiv:1703.09439*.
- Mathew, A., Amudha, P., & Sivakumari, S. 2020. Deep learning techniques: an overview. Singapore: Springer.
- Mathew, A., Amudha, P. & Sivakumari, S. 2020. Deep learning techniques: an overview. In *International Conference on Advanced Machine Learning Technologies and Applications*. pp. 599–608. Springer.
- Mehri, S., Kumar, K., Gulrajani, I., Kumar, R., Jain, S., Sotelo, J., Courville, A. & Bengio, Y. 2016. SampleRNN: An unconditional end-to-end neural audio generation model. *ArXiv Preprint ArXiv:1612.07837*.
- Mesnil, G., Dauphin, Y., Yao, K., Bengio, Y., Deng, L., Hakkani-Tur, D., He, X., Heck, L., Tur, G. & Yu, D. 2014. Using recurrent neural networks for slot filling in spoken language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 23(3): 530–539.
- Mohasi, L. & Mashao, D. 2005. Phonetization for text-to-speech synthesis in Sesotho. In *Sixteenth Annual Symposium of the Pattern Recognition Association of South Africa*. p. 121. Citeseer.
- Mousa, A.E.-D. & Schuller, B. 2016. Deep bidirectional long short-term memory recurrent neural networks for grapheme-to-phoneme conversion utilizing complex many-to-many alignments.
- Nawar Halabi. 2016. Modern Standard Arabic Phonetics for Speech Synthesis. UNIVERSITY OF SOUTHAMPTON.
- Pan, S.J. & Yang, Q. 2010. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering* 22(10): 1345–1359.
- Parikh, A.P., Täckström, O., Das, D. & Uszkoreit, J. 2016. A decomposable attention model for natural language inference. *ArXiv Preprint ArXiv:1606.01933*.
- Pascanu, R., Mikolov, T. & Bengio, Y. 2013. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*. pp. 1310–1318. PMLR.

- ProgrammableWeb Staff. 2020. 9 Top Text to Speech APIs
<https://www.programmableweb.com/news/9-top-text-to-speech-apis/brief/2020/06/07> [18 July 2022].
- Rahate, P.M. & Chandak, M. (n.d.). An Experimental Technique on Text Normalization and its Role in Speech Synthesis.
- Roark, B., Sproat, R., Allauzen, C., Riley, M., Sorensen, J. & Tai, T. 2012. The OpenGrm open-source finite-state grammar software libraries. In *Proceedings of the ACL 2012 System Demonstrations*. pp. 61–66.
- Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A. & Bernstein, M. 2015. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision* 115(3): 211–252.
- Ruzsics, T. & Samardžić, T. 2019. Multilevel Text Normalization with Sequence-to-Sequence Networks and Multisource Learning. *ArXiv Preprint ArXiv:1903.11340*.
- Schroff, F., Kalenichenko, D. & Philbin, J. 2015. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pp. 815–823.
- Schuster, M. & Paliwal, K.K. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing* 45(11): 2673–2681.
- Shang, W., Chiu, J. & Sohn, K. 2017. Exploring normalization in deep residual networks with concatenated rectified linear units. In *Thirty-First AAAI Conference on Artificial Intelligence*.
- Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G. & Dean, J. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *ArXiv Preprint ArXiv:1701.06538*.
- Sotelo, J., Mehri, S., Kumar, K., Santos, J.F., Kastner, K., Courville, A. & Bengio, Y. 2017. Char2wav: End-to-end speech synthesis.
- Sproat, R., Black, A.W., Chen, S., Kumar, S., Ostendorf, M. & Richards, C. 2001. Normalization of non-standard words. *Computer Speech & Language* 15(3): 287–333.
- Sproat, R. & Gorman, K. 2018. A brief summary of the Kaggle text normalization challenge. *Medium Blog Post*.
- Sproat, R. & Jaitly, N. 2016. RNN approaches to text normalization: A challenge. *ArXiv Preprint ArXiv:1611.00068*.
- Srivastava, R.K., Greff, K. & Schmidhuber, J. 2015. Highway networks. *ArXiv Preprint ArXiv:1505.00387*.
- Sun, H., Tan, X., Gan, J.-W., Liu, H., Zhao, S., Qin, T. & Liu, T.-Y. 2019. Token-level ensemble distillation for grapheme-to-phoneme conversion. *ArXiv Preprint ArXiv:1904.03446*.

- Sutskever, I., Vinyals, O. & Le, Q. V. 2014. Sequence to sequence learning with neural networks. *Advances in Neural Information Processing Systems 27*.
- Tang, Z., Shi, Y., Wang, D., Feng, Y. & Zhang, S. 2017. Memory visualization for gated recurrent neural networks in speech recognition. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. pp. 2736–2740. IEEE.
- Taylor, P. 2009. *Text-to-speech synthesis*. Cambridge university press.
- Toshihiro Takahashi. 2018. Statistical max pooling with deep learning. US Patent.
- Toshniwal, S. & Livescu, K. 2016. Jointly learning to align and convert graphemes to phonemes with neural attention models. In *2016 IEEE Spoken Language Technology Workshop (SLT)*. pp. 76–82. IEEE.
- Tran, D.C. 2020. The first vietnamese f0sd-tacotron-2-based text-to-speech model dataset. *Data in Brief 31*.
- Vinyals, O., Kaiser, Ł., Koo, T., Petrov, S., Sutskever, I. & Hinton, G. 2015. Grammar as a foreign language. *Advances in Neural Information Processing Systems 28*.
- Vu, N.T., Gupta, P., Adel, H. & Schütze, H. 2016. Bi-directional recurrent neural network with ranking loss for spoken language understanding. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. pp. 6060–6064. IEEE.
- Wang, W., Xu, S. & Xu, B. 2016. First Step Towards End-to-End Parametric TTS Synthesis: Generating Spectral Parameters with Neural Attention. In *Interspeech*. pp. 2243–2247.
- Wang, Y., Skerry-Ryan, R.J., Stanton, D., Wu, Y., Weiss, R.J., Jaitly, N., Yang, Z., Xiao, Y., Chen, Z. & Bengio, S. 2017. Tacotron: Towards end-to-end speech synthesis. *ArXiv Preprint ArXiv:1703.10135*.
- Watanabe, S., Hori, T., Karita, S., Hayashi, T., Nishitoba, J., Unno, Y., Soplín, N.E.Y., Heymann, J., Wiesner, M. & Chen, N. 2018. Espnet: End-to-end speech processing toolkit. *ArXiv Preprint ArXiv:1804.00015*.
- Weiss, R.J., Skerry-Ryan, R.J., Battenberg, E., Miaooryad, S. & Kingma, D.P. 2021. Wave-tacotron: Spectrogram-free end-to-end text-to-speech synthesis. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. pp. 5679–5683. IEEE.
- Wu, Y., Schuster, M., Chen, Z., Le, Q. V, Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q. & Macherey, K. 2016. Google’s neural machine translation system: Bridging the gap between human and machine translation. *ArXiv Preprint ArXiv:1609.08144*.
- Yamashita, R., Nishio, M., Do, R.K.G. & Togashi, K. 2018. Convolutional neural networks: an overview and application in radiology. *Insights into Imaging 9(4)*: 611–629.

- Yu, D., Xiong, W., Droppo, J., Stolcke, A., Ye, G., Li, J. & Zweig, G. 2016. Deep Convolutional Neural Networks with Layer-Wise Context Expansion and Attention. In *Interspeech*. pp. 17–21.
- Yu, J., Xie, L., Xiao, X. & Chng, E.S. 2017. An end-to-end neural network approach to story segmentation. In *2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. pp. 171–176. IEEE.
- Zen, H., Tokuda, K. & Black, A.W. 2009. Statistical parametric speech synthesis. *Speech Communication* 51(11): 1039–1064.
- Zhang, H., Sproat, R., Ng, A.H., Stahlberg, F., Peng, X., Gorman, K. & Roark, B. 2019. Neural models of text normalization for speech applications. *Computational Linguistics* 45(2): 293–337.
- Zhang, S., Lei, M. & Yan, Z. 2019. Investigation of Transformer Based Spelling Correction Model for CTC-Based End-to-End Mandarin Speech Recognition. In *Interspeech*. pp. 2180–2184.
- Zhou, P., Qi, Z., Zheng, S., Xu, J., Bao, H. & Xu, B. 2016. Text classification improved by integrating bidirectional LSTM with two-dimensional max pooling. *ArXiv Preprint ArXiv:1611.06639*.
- Zhou, Y., Wang, H., Xu, F. & Jin, Y.-Q. 2016. Polarimetric SAR image classification using deep convolutional neural networks. *IEEE Geoscience and Remote Sensing Letters* 13(12): 1935–1939.