AI Virtual Voice Assistant

Abdelrahman Amr 193159 | Alaa Amgad 193965 | Gehad Ihab 190561 | Omar Ashraf 190265 | Ziad Hegazi 195407

Course Project: MSE 551

Mechatronics Systems Department, Faculty of Engineering, MSA University

Fall 2022

Abstract- **Artificial intelligence technology is fast becoming the scope of every major research and industrial field. The use of this technology has given way to a number of assistive programs that make everyday tasks easier and more convenient. One such application of AI-based assistive technologies is the rise of the virtual voice assistant. The development of voice assistants made the luxury of hands-free device control a daily reality. Whether it is embedded in one's phone like Bixby and Siri or sitting at a desk like Alexa and Google Home, it enables the user to perform tasks such as turning on the lights, receiving answers to a question, playing music, or placing an online order, this technology is able both to respond to a user's command and anticipate their every need.**

1. Introduction

Voice assistants are devices that use voice recognition technology, AI to respond to humans, natural language processing, and voice synthesis to listen to specific voice commands and return relevant information or perform specific functions as requested by the user.

Voice assistant software can be found on smart speakers, smartwatches, mobile phones, tablets, and other devices. The most well-known are Alexa, Siri, and Google Assistant which are taking over our homes thanks to their compatibility with washing machines, light bulbs, ovens, air conditioning units, etc.

While voice assistants may be entirely a software system on which all devices may be combined, other assistants are specifically designed for each individual device use, such as the Amazon Alexa clock.

The cutting-edge method of human-system interface is voice instructions that can be easily entered into a machine.

Understanding the significance of this, we created a system that can be installed anyplace nearby and that you can ask to assist you with any task just by speaking. With continuous updates and reminders, this device may be highly useful for daily usage and improve your productivity. Why could we require it? because using your voice instead of a traditional enter key is becoming more and more popular.

2. Problem Definition

Everyday tasks are quickly becoming easier and less time-consuming due to the emergence of various smart technologies that aid in the automation of many applications. Accessibility for differently-abled people remains an area that can see large improvements. One such technology that can aid this particular group of people is the virtual voice assistant which has become very prominent in recent years as the availability of smartphones skyrocketed. For the majority of its applications, however, computational power and a solid background in machine learning are vital requirements for building similar technology.

3. Literature Survey

As summarized by [1] software agents known as voice assistants can understand spoken language and respond with synthetic voices. The most well-known voice assistants are Google Assistant, Apple's Siri, Amazon's Alexa, and Microsoft's Cortana, and they are all built into smartphones or specific home speakers. Users can

use voice commands to manage other common chores like email, to-do lists, and calendars in addition to asking their assistants questions, controlling home automation devices, and controlling media playback.

The authors in [2] propose a two-way, automated, and agnostic approach to fill this theoretical and methodological gap. Using Amazon Alexa as the platform for the AI voice assistant, a "Proof of Concept" prototype was created to evaluate the viability. The result demonstrates the validity of both the created and retrieved information. The components of the suggested solution are also highly interoperable, including the AI voice assistant interface and the mediation environment used to translate spoken requests and obtain data into CSV files.

Moreover, the authors in [3] created a model to ascertain whether a one-second audio sample contains a specific word (from a set of ten), an unidentified word, or silence. The TensorFlow Speech Recognition tutorial's recommended CNN, a low-latency CNN, and an adversarially trained CNN are the models that need to be put into practice. As a result, it is shown how to transfer an issue with audio identification to the more well-researched field of image classification, where the potent convolutional neural network techniques are completely developed. Additionally, they illustrate the Virtual Adversarial Training (VAT) technique's relevance to this problem domain, serving as a potent regularizer with a range of potential future applications.

[4] put forth a brand-new, user-defined, end-to-end keyword-detecting technique that makes advantage of linguistically similar patterns in speech and text sequences. Our method compares input queries with an enrolled text keyword sequence, unlike other techniques that required voice keyword enrolment. Using an attention-based cross-modal matching strategy that is trained end-to-end with monotonic matching loss and keyword classification loss, the audio and text representations were arranged within a single latent space. Comparing their suggested strategy to previous single-modal and cross-modal baselines, it produced findings that are competitive on a variety of assessment sets including the Qualcomm dataset hat will be used for the proposed model.

[5] first speaks about the uses of AI assistants and how they can be utilized to speed up everyday routines. The authors then go into details about the system they propose and the features it has such as staying on standby until called by the predefined function they created, it then searches the web for whatever parameter is spoken by the user, and prints the output. They used a GUI so the user would be able to interact with the system and be provided with the necessary information. Even though they displayed the project with an eye-catching GUI it seems to lack features since all that can be done is search the web.

[6] proposed a method to overcome the intrinsic tradeoff between temporal and frequency precision that occurs for spectrum representations by using convolutional filters. On a difficult internal speech test set, the paper also finds more useful representations by simultaneously learning at several scales, which leads to a 20.7% overall reduction in word mistake rate when compared to networks with the same number of parameters trained on spectrograms. Also, it seems learning convolutional filters overcomes spectrogram representations, but many cutting-edge systems today train on clusters of GPUs since training on individual GPUs, where memory is expensive, can be excessively slow. This is particularly troublesome when training on lengthy utterances since an increase in memory is needed to save all the activations as the number of filters and stride lengthen.

In this study [7], a unique CNN design called SincNet is proposed, which stimulates the first convolutional layer to find deeper filters. This offers a very effective and condensed way for building a unique filter bank that is specifically suited for the application. This paper research’ speech recognition and speaker verification tasks show that the recommended architecture outperforms a traditional CNN on raw waveforms and converges more quickly. Beyond performance gains, SincNet greatly outperforms a regular CNN in terms of convergence speed and is more computationally economical because to the use of filter symmetry.

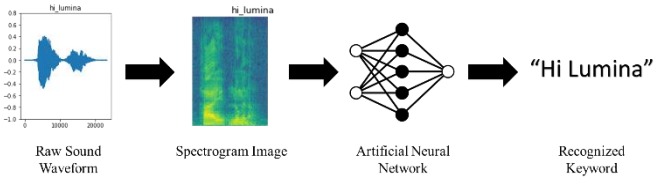
4. Proposed Solution and Methodology

With a virtual voice assistant, the user will be able to control their device to its fullest; they will be able to play music, open YouTube videos, and even search on the internet with just their voice. This encapsulates the

main focus of artificial intelligence which is to work alongside humans to make their lives easier by automating some of their daily activities and organizing their work in the same way a human assistant would. The assistant must be able to understand human language and be able to provide responses based on its internal speech recognition system.

Based on the survey conducted on similar work, and based on computational limitations, the proposed project will use a keyword-spotting machine learning algorithm instead of a full-scale speech recognition model. Such high-level models use what is known as a Recurrent Neural Network (RNN) which on top of being computationally demanding, is also rather difficult to adapt for the proposed application.

The model that will be built and used will convert the raw sound file waveforms into corresponding spectrogram images to extract features. The spectrogram will then be employed as the input to a convolutional Neural Network (CNN). The model will use TensorFlow and the Keras deep-learning API to build and train the model on the selected dataset.



The use of spectrograms versus the usage of raw audio data for speech recognition purposes has always been a subject of research and extensive trials. Although both approaches are applicable and can yield great results many machine learning experts lean towards the use of the spectrogram image over the raw audio data for several reasons including

- Computational Power/Time is almost always smaller when employing spectrograms and 2D convolutional layers. Audio waveforms often require

5. Dataset and Evaluation Metrics

The dataset used was the Qualcomm Keyword Speech Dataset. The dataset has 4,270 utterances of four English keywords spoken by 50 people. The four keywords are Hey Android, Hey Snapdragon, Hi Galaxy, and Hi Lumina. The following table shows the details for each keyword. Each wave file has been recorded with a sampling rate of 16 kHz, mono channel, and 16 bits bit-depth.

As for the metrics of evaluation, the selected criteria were accuracy, loss or error, validation accuracy, and validation loss for the model training and fitting. In addition, the computational time is also an important criterion to monitor and analyze, as the main aim of the model is to be used in real-time applications. The metrics are defined as follows:

- **Accuracy:** How close the predicted value is to the target output. a qualitative presentation of how close the model output is to the labeled data points.

$$Accuracy = \frac{T_P + T_N}{T_P + F_P + T_N + F_N}$$

		Predicted 0	Predicted 1
Actual	0	T_N	F_P
	1	F_N	T_P

T_N = True Negative
F_N = False Negative
F_P = False Positive
T_P = True Positive

- **Loss:** Also referred to as error, is the difference in value between the target and predicted quantities. Model weights are adjusted so that the loss value is at a minimum.

$$CE = -\sum_{i=1}^{i=N} y_{true_i} \cdot \log\left(y_{pred_i}\right)$$

$$CE = -\sum_{i=1}^{i=N} y_i \cdot \log\left(\hat{y}_i\right)$$

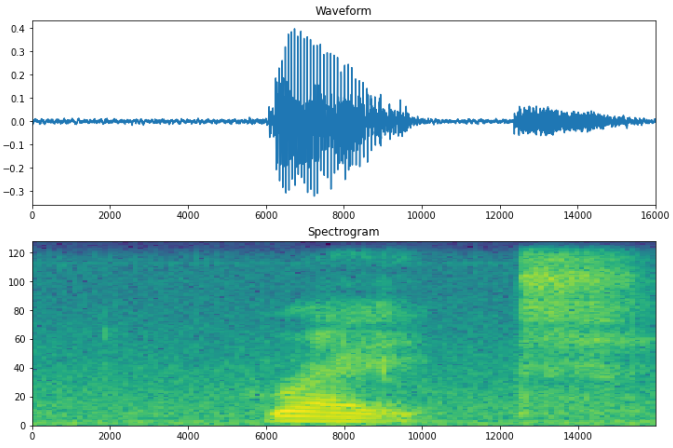
- **Validation Accuracy:** Indicates the accuracy of the predictions of a randomly split validation set after each epoch.
- **Validation Loss:** A metric that indicates the loss percentage on the validation set.

6. Experimental Setup

The model will be developed using Google Colaboratory. The algorithm begins by importing the necessary libraries and modules, namely TensorFlow. The next necessary step is to import and split the dataset into training, validation, and testing sets. The split ratio was selected to be 80:10:10 for the training, validation, and testing respectively.

Reading the audio files and decoding them is the next step in the dataset setup. This is necessary because ".wav" audio files are initially read as binary files which will need to be converted into numerical tensors and normalized.

Now for the most vital step, obtaining the spectrogram image of each sound waveform. A spectrogram is a 2D representation of the frequency changes over time. This is done by applying what is known as a Short-Time Fourier Transform (STFT) to convert the audio into the time-frequency domain. It is important to note that STFT produces an array of complex numbers representing magnitude and phase, but for the proposed setup only the magnitude data will be used. The figure below shows a waveform with its corresponding spectrogram.



As mentioned before, the model will use a simple CNN that will take the spectrogram image of the audio as an input. The model additionally includes the following preprocessing layers:

To enable the model to train more quickly, a resizing layer downsamples the input and a normalization layer that uses the image's mean and standard deviation to normalize each pixel.

The full model has the following characteristics:

- Model was built sequentially with 2 convolutional layers, a max-pooling layer, flattening, and dense layers as shown.
- Input layer was used to resize all spectrogram images to uniform dimensions.
- Dropout layers were used to prevent model overfitting.

The model parameters are summarized in the figure below.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resizing (Resizing)         (None, 32, 32, 1)         0

 normalization_1 (Normalizat  (None, 32, 32, 1)        3
 ion)

 conv2d (Conv2D)             (None, 30, 30, 32)        320

 conv2d_1 (Conv2D)           (None, 28, 28, 64)        18496

 max_pooling2d (MaxPooling2D  (None, 14, 14, 64)       0
 )

 dropout (Dropout)           (None, 14, 14, 64)        0

 flatten (Flatten)           (None, 12544)             0

 dense (Dense)               (None, 128)               1605760

 dropout_1 (Dropout)         (None, 128)               0

 dense_1 (Dense)             (None, 4)                 516

=================================================================
Total params: 1,625,095
Trainable params: 1,625,092
Non-trainable params: 3
_________________________________________________________________
```

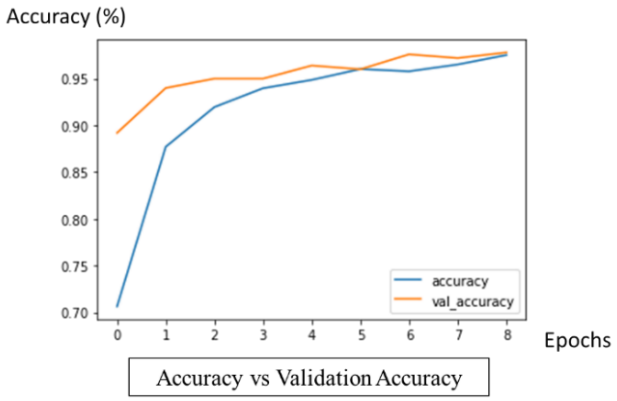
7. Results

For the execution time, it was rather slow (4h 31m) due to the pre-processing time for converting each sound file to a spectrogram image. Model fitting time, however, was reasonable at about 11s per epoch.

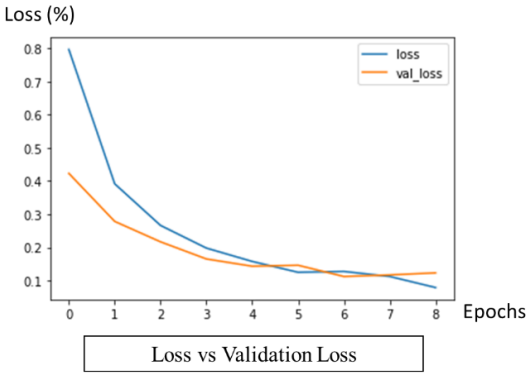
The accuracy, loss, validation-loss, and validation accuracy after the nine performed epochs are shown in the model runtime figure below.

```
loss: 0.7960 - accuracy: 0.7064 - val_loss: 0.4230 - val_accuracy: 0.8920
- loss: 0.3916 - accuracy: 0.8771 - val_loss: 0.2784 - val_accuracy: 0.9400
- loss: 0.2663 - accuracy: 0.9196 - val_loss: 0.2169 - val_accuracy: 0.9500
- loss: 0.1980 - accuracy: 0.9398 - val_loss: 0.1653 - val_accuracy: 0.9500
- loss: 0.1578 - accuracy: 0.9486 - val_loss: 0.1434 - val_accuracy: 0.9640
- loss: 0.1251 - accuracy: 0.9602 - val_loss: 0.1463 - val_accuracy: 0.9600
- loss: 0.1277 - accuracy: 0.9578 - val_loss: 0.1125 - val_accuracy: 0.9760
- loss: 0.1126 - accuracy: 0.9651 - val_loss: 0.1172 - val_accuracy: 0.9720
- loss: 0.0792 - accuracy: 0.9752 - val_loss: 0.1234 - val_accuracy: 0.9780
```

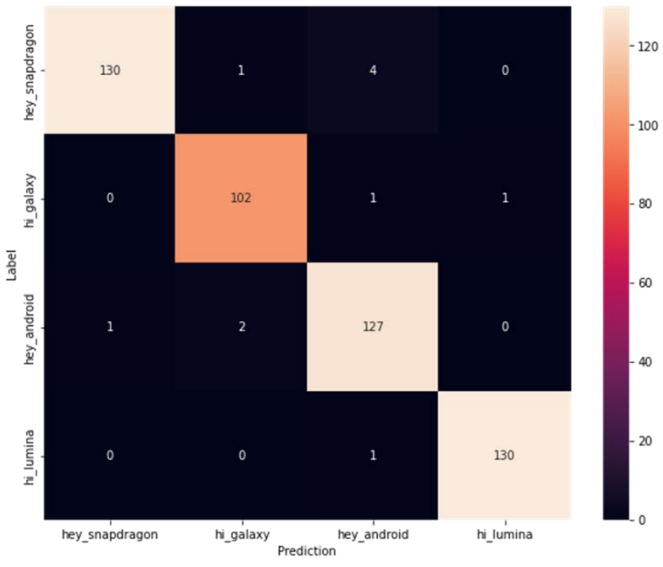
The following graphs present how the evaluation metrics changed over the training epochs. Moreover, the confusion matrix shows the corresponding number of predictions for each keyword label.



Accuracy vs Validation Accuracy



Loss vs Validation Loss



8. Model Analysis and Comparison

Comparing our results with [4] on the same dataset, it can be seen that the proposed model provides a higher accuracy at 97.52% compared to 94.51% and a lower error rate (loss) at 7.92% compared to 12.15%.

Method	EER (%)				AUC (%)		
	G	Q	LP_E	LP_H	G	Q	LP_E
(I) CTC [5]	31.65	18.23	14.67	35.22	66.36	89.69	92.29
(I) Attention [6]	**14.75**	49.13	28.74	41.95	**92.09**	50.13	78.74
(II) Triplet [7]	35.60	38.72	32.75	44.36	71.48	66.44	63.53
(II) Proposed	27.25	**12.15**	**8.42**	**32.90**	81.06	**94.51**	**96.70**

	Loss (Error)(%)	Accuracy(%)
Proposed	7.92%	97.52%

9. Voice Assistant Implementation

Voice Assistant is a virtual assistant that uses speech recognition, natural language processing, and speech synthesis to take actions to help its users' needs. After creating the speech recognition model, we developed the system's ability to use that model to process what was said and respond accordingly.

The libraries in the red boxes are what we used in our voice assistant features; webbrowser and wikipedia libraries are used to search for information on the internet and retrieve it to the user, wolframalpha is used for executing mathematical calculations, and lastly, the pywhatkit sends messages to single or group chats on Whatsapp as well as, used to surf the internet

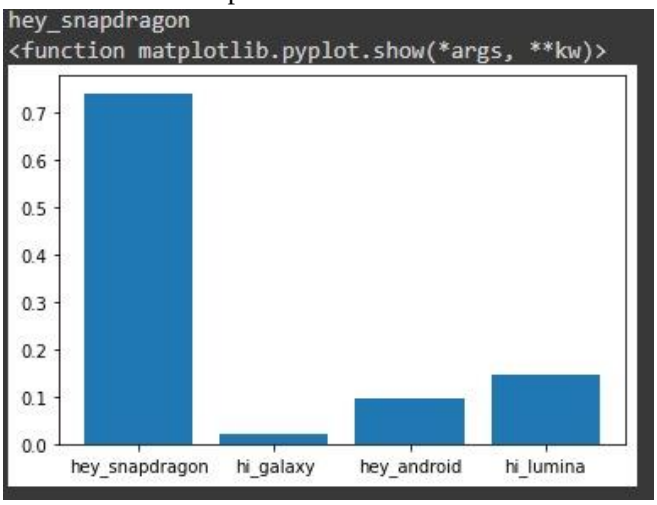
```
import os
import pathlib
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import tensorflow as tf
from tensorflow.keras.layers.experimental import preprocessing
from tensorflow.keras import layers
from tensorflow.keras import models
from IPython import display
from IPython.display import Javascript
from google.colab import output
from base64 import b64decode
import wave
import soundfile as sf
import librosa
from datetime import datetime
from logging.config import listen
import speech_recognition as sr
import pyttsx3
import webbrowser
import wikipedia
import wolframalpha
import pyaudio
from io import BytesIO
from pydub import AudioSegment
from gtts import gTTS
from IPython.display import Audio
import pywhatkit as kt
```

10. Real-Time Deployment Results

Waiting for the model to train and then proceeding with giving us a prediction is both time and resource-consuming. Therefore, we create a real-time model that needs to be trained only once and then used anytime to provide a prediction.

To ensure we maintained good accuracy for valid prediction we tested to see the model's behavior after deployment. Below we can see a screenshot of one of the tests made; inspecting the output given for an input

multiple times and seeing how many times it makes a correct prediction, the results were promising since around 75% of the predictions were the correct ones.



Here we gave the system input and checked to see all the possible outcomes to check how close the prediction was.

```
Listening for a command
Recognizing speech...
result2:
{   'alternative': [   {   'confidence': 0.91929084,
                           'transcript': 'system say hello'},
                       {'transcript': 'systems say hello'},
                       {'transcript': 'the system say hello'},
                       {'transcript': 'a system say hello'},
                       {'transcript': 'systemc hello'}],
    'final': True}
The input speech was: system say hello
```

11.Limitations and Future Work

The proposed model proved to be accurate during its training, validation, and testing phases but was not as robust as we would have liked during its real-world deployment. First of all, the model is affected heavily by changes in audio sample rates, for example the dataset used had all audio files at 16kHz sample and the microphone input had a sample rate of 48kHz, due to this the predictions from the model were often inaccurate although normalization attempts improved the results slightly.

Secondly, the model as developed using google colab which is an online compiler that does not have local device privileges, this meant that for the voice assistant to be able to manipulate local applications, another environment had to be employed, this was chosen to be Jupyter Notebook.

As for the future development, it is greatly encouraged to solve the issues mentioned above and to create a more complete experience for the user. This can be done by expanding the model to discern a larger set of keywords or even creating a full speech recognition model. Another area that can see much improvement is the GUI which was not developed in this project but would be a great addition especially if the system was to be deployed as readily available programs for users to run.

12.List of Contributions

The proposed project contributions are summarized as follows:

- Use of a TensorFlow end-to-end model instead of a pre-trained model which can be difficult to adapt.
- The use of dropout layers to prevent overfitting.
- Adapting the trained model for real-time keyword recognition.
- Setting seed value to ensure reproducibility of results.
- Higher validation accuracy at 97.8% after just 9 epochs.

13. Conclusion

All in all, the potential of voice assistance is high with an ongoing evolution in the future. Developing an AI assistant proved to be a challenging task both computationally and algorithmically. We created a keyword recognition model and then used it to implement an AI virtual assistant that executes specific commands such as surfing the web and extracting relevant information. Even though it was rewarding to reach that far in our research, this is not the limit to what an AI assistant can undertake, and it can be worked on in the foreseeable future.

14. Future Work

The upcoming work for the rest of the project:

- Using the model with real-time microphone input from the user.
- Using keyword spotting technology to implement assistant responses to perform certain tasks.
- To test and evaluate the real-time performance of the virtual assistant.

15. References

[1] M. B. Hoy, "Alexa, Siri, Cortana, and More: An Introduction to Voice Assistants," *Medical Reference Services Quarterly,* vol. 37, no. 1, pp. 81-88, 12 January 2018.

[2] F. Elghaish, J. K. Chauhan, S. Matarneh, F. P. Rahimian and M. R. Hosseini, "Artificial intelligence-based voice assistant for BIM data management," *Automation in Construction,* vol. 140, 13 May 2022.

[3] S. K. Gouda, S. Kanetkar, V. Harrison and M. K. Warmuth, "Speech Recognition: Key Word Spotting through Image Recognition," University of California, Santa Cruz, 2020.

[4] H.-K. Shin, . H. Han, D. Kim, . S.-W. Chung and . H.-G. Kang, "Learning Audio-Text Agreement for Open-vocabulary Keyword Spotting," in *Interspeech 2022*, Incheon, Korea, 2022.

[5] G. Preethi, K. Abishek, S. Thiruppugal and D. A. Vishwaa , "Voice Assistant using Artificial Intelligence," *International Journal of Engineering Research & Technology (IJERT),* vol. 2, no. 5, pp. 453-457, 2022.

[6] Zhu, Z., Engel, J. H., & Hannun, A. (2016). Learning Multiscale Features Directly from Waveforms. Interspeech 2016. https://doi.org/10.21437/interspeech.2016-256

[7] M. Ravanelli and Y. Bengio, "Speaker Recognition from Raw Waveform with SincNet," 2018 IEEE Spoken Language Technology Workshop (SLT), 2018, pp. 1021-1028, doi: 10.1109/SLT.2018.8639585.