Building and Benchmarking an Arabic Speech Commands Dataset for Small-Footprint Keyword Spotting

Preprint · April 2021

CITATIONS
0

READS
659

3 authors:

Abdulkader Ghandoura
Technische Universität München
2 PUBLICATIONS **3** CITATIONS
SEE PROFILE

Farouk Hjabo
2 PUBLICATIONS **3** CITATIONS
SEE PROFILE

Oumayma Dakkak
Higher Institute for Applied Sciences and Technology
34 PUBLICATIONS **70** CITATIONS
SEE PROFILE

Building and Benchmarking an Arabic Speech Commands Dataset for Small-Footprint Keyword Spotting*

Abdulkader Ghandoura[a,*], Farouk Hjabo[b], Oumayma Al Dakkak[a,c]

[a] *Syrian Virtual University, Damascus, Syria.*
[b] *Innopolis University, Innopolis, Russia.*
[c] *Higher Institute for Applied Sciences and Technology, Damascus, Syria.*

Abstract

The introduction of the Google Speech Commands dataset accelerated research and resulted in a variety of new deep learning approaches that address keyword spotting tasks. The main contribution of this work is the building of an Arabic Speech Commands dataset, a counterpart to Google's dataset. Our dataset consists of 12000 instances, collected from 30 contributors, and grouped into 40 keywords. We also report different experiments to benchmark this dataset using classical machine learning and deep learning approaches, the best of which is a Convolutional Neural Network with Mel-Frequency Cepstral Coefficients that achieved an accuracy of ~98%. Additionally, we point out some key ideas to be considered in such tasks.

Keywords: Arabic Speech Dataset, Speech Recognition, Keyword Spotting, Machine Learning, Deep Learning

1. Introduction

Automatic speech recognition (ASR) is a growing area of interest as people are gaining more access to smart devices. The keyword spotting (KWS) task, in contrast to Large Vocabulary Continuous Speech Recognition (LVCSR) tasks, is concerned with the detection of a relatively small set of predefined keywords. Although one might think that LVCSR subsumes KWS, there are subtle differences that should be considered.

KWS systems are found on mobile devices. For example, saying 'Ok Google' or 'Hey Siri' can trigger Google Assistant or Apple's Siri, respectively. KWS can also be found on other systems like smart homes, Internet of Things (IoT) devices, or virtual assistant devices like Amazon's Alexa. Therefore, to be able to run on such systems with low resources and limited battery, KWS systems need to have a small footprint and be as computationally efficient as possible. Especially that these systems need to run as background processes receiving a stream of audio.

*https://doi.org/10.1016/j.engappai.2021.104267
*Corresponding author
Email addresses: abdulkader_75241@svuonline.org, f.hjabo@innopolis.university, oumayma.dakkak@hiast.edu.sy

In contrast, advanced ASR systems require more computational resources. They usually run on the cloud (Tang and Lin, 2018). It is impractical to do this for KWS, as this is more costly and raises additional problems like network latency and privacy concerns. Many voice interfaces rely on KWS to initiate interactions, and only then, it is possible to send the audio to a remote web service. This discussion suggests that a KWS system must respect the following constraints (Warden, 2018):

- It must have a small-footprint, i.e., it must be small and run efficiently in terms of memory, computation, and power consumption.
- The input is mostly silence, background noise, or random utterances, so false positives must be minimized, i.e., the application must not be triggered by a noise or random speech.
- The significant part of an input speech signal is a single word or a short phrase, not an entire sentence.

With the availability of a suitable dataset, it is possible to train different kinds of models to detect an application-based predefined set of keywords. The Google Speech Commands (GSC) dataset was built for such purposes (Warden, 2018), and it has been used by many researchers as the associated paper (Warden, 2018) was cited 320 times at the time of this writing. GSC is available in two versions: version 0.01 was released on August 3$^{\text{rd}}$, 2017, and version 0.02 was released on April 11$^{\text{th}}$, 2018. We built the Arabic Speech Commands (ASC) dataset to support similar purposes. The publication of the ASC dataset also serves to enrich the Arabic digital content that is available online.

The publication of such datasets has great advantages for the machine learning/deep learning community. In fact, the availability of multiple computer vision datasets encouraged research and collaboration, thus allowing different comparisons among many approaches (Warden, 2018). Most notably, the publication of ImageNet (Deng et al., 2009), which launched the ImageNet Large Scale Visual Recognition Challenge, helped in radical developments in the computer vision field.

In addition, a KWS dataset serves educational purposes in the speech recognition domain, as most tasks are usually complicated involving different steps in a complex pipeline; Generally, advanced ASR systems require knowledge in speech signal processing and the development of acoustic and language models.

2. Related Work

With the recent advances in deep learning, it is possible to directly use Neural Networks (NNs) in an end-to-end fashion by feeding them raw waveform signals (e.g., Graves, 2012). However, this is still challenging and requires high computational cost, and even the performance of such models tends to be worse than the ones that use conventional preprocessing methods (Graves and Jaitly, 2014). Spectrograms can be used instead as a different and more convenient representation (e.g., Graves and Jaitly, 2014). However, for a KWS task, it is better to use some well-established feature extraction algorithm as a preprocessing step to

keep our whole pipeline more efficient. Accordingly, Mel-Frequency Cepstral Coefficients (MFCCs) (Huang et al., 2001, Sec. 6.5.2) are usually used as a minimal preprocessing step (e.g., Chen et al., 2014; Sainath and Parada, 2015; Majumdar and Ginsburg, 2020).

Most of the recent related works that address KWS are based on the GSC dataset. Other datasets that support general ASR tasks (e.g., Mozilla's Common Voice dataset[1]) are not easily adaptable to keyword spotting (Warden, 2018). Earlier approaches used hybrid models of Neural Networks and Hidden Markov Models (HMMs) (e.g., Bengio et al., 1992), whereas recent approaches rely solely on NNs, i.e., deep learning.

Chen et al. (2014) investigated the application of Deep Neural Networks (DNNs), also known as Multi-Layer Perceptrons (MLPs). Sainath and Parada (2015) investigated Convolutional Neural Networks (CNNs), which proved to be much more effective. To keep track of the footprint of a model, Sainath and Parada (2015) suggested keeping track of two metrics: the number of parameters in the model and the number of multiplications performed by the model. We can see that a lower number of parameters corresponds to lower memory resources needed and that a lower multiplications number corresponds to lower computational resources needed. Sainath and Parada (2015) investigated and compared several configurations that correspond to different footprints.

Newer more advanced approaches adopted more complex architectures. Tang and Lin (2018) studied the application of Residual Networks and dilated convolutions. Wang et al. (2018) worked on Gated Convolutional Long Short-Term Memory (LSTM) models. de Andrade et al. (2018) investigated convolutions and LSTMs with the addition of an attention layer. And most recently, Majumdar and Ginsburg (2020) used a deep residual neural network of multiple blocks, which they refer to as MatchboxNet. Majumdar and Ginsburg (2020) also showed how data augmentation using an auxiliary noise dataset improves the robustness of the model in the presence of background noise. In addition, McMahan and Rao (2017) showed the improvement in speech commands recognition attainable by transfer learning using pre-trained models on a different audio classification task.

Lastly, Benamer and Alkishriwo (2020) very recently did a similar work to ours. Unfortunately, we only learned about them after the acceptance of this paper.

3. Dataset Description

This section describes how the ASC dataset was collected, verified, and organized. We try to compare its properties with that of the GSC dataset. The dataset is publicly available online[2] under the Creative Commons Attribution 4.0 International (CC BY 4.0) license[3].

[1] `https://commonvoice.mozilla.org/en`

[2] `http://doi.org/10.5281/zenodo.4662481`

[3] `https://creativecommons.org/licenses/by/4.0`

Eventually, our dataset is a list of pairs (x, y) where x is the input speech signal, and y is the corresponding keyword. The final dataset consists of 12000 such pairs, comprising 40 keywords. Each audio file is one second in length sampled at 16 kHz. We have 30 participants, each of them recorded 10 utterances for each keyword. Therefore, we have 300 audio files for each keyword in total ($30 \times 10 \times 40 = 12000$). Lastly, the total size of the dataset is ~384 MB.

3.1. Criteria & Properties

Since it is not acceptable to ask any contributor to do a re-recording, our first step in the data collection process was to define the properties of the audio file to do the recordings properly. Accordingly, we chose the following settings for the audio files: a sampling rate of 16 kHz as we found it to be the most common; 16 bits to represent each sample in the signal; one audio channel, i.e., mono-signal, as opposed to stereo-signal. We also set the length of each audio file to be one second, and we used the `.wav` extension to save the audio files, similar to the GSC dataset.

3.2. Word Choice

We chose a set of common words that can be used as speech commands to control some applications and devices such as a simple photo browser or a keypad. Some of these words (e.g., digits and directions) were inspired by GSC, while the rest of them are chosen so that they can be grouped into broad useful, possibly intersecting, categories. Table 1 lists the 40 chosen keywords with their translations into Arabic and pronunciations[4] in the International Phonetic Alphabet (IPA). For example, we can choose a subset of these keywords to use as commands for a small robot: start, stop, forward, backward, right, left, up, down, enable, disable, etc.

3.3. Collection Procedure

Initially, we estimated the approximate time to collect the recordings from each contributor without annoying or fatiguing them. We found it acceptable to record each keyword ten times by each contributor.

As a preliminary step, we asked some contributors to record a single keyword ten times in a row, they unintentionally used almost the same pitch. If we used this method, we would lose the chance of getting a variety of utterances for every single keyword recorded by the same contributor. On the other hand, explicitly asking them to try changing their pitch in every repetition confused most of them (especially the elderly) and led many contributors to pause longer or use unnatural pitches, which would lead to a longer time in collecting data and the presence of unrealistic utterances. Therefore, we asked them to pronounce all of the 40 keywords, one by one consecutively, and we refer to this procedure as a *round* which is performed

[4]Adapted from `https://dict.com/arabic-english`

Translation	Pronunciation	Keyword	Translation	Pronunciation	Keyword
Zero	sˤifr	صِفر	Enable	tafʕiːl	تفعيل
One	waːħid	واحد	Disable	taʕtˤiːl	تعطيل
Two	iθnaːn	اثنان	Ok	muwaːfiq	موافق
Three	θalaːθa	ثلاثة	Cancel	ʔilɣaːʔ	إلغاء
Four	ʔarbaʕa	أربعة	Open	fatħ	فتح
Five	xamsa	خمسة	Close	ʔiɣlaːq	إغلاق
Six	sitta	ستة	Zoom in	takbiːr	تكبير
Seven	sabʕa	سبعة	Zoom Out	tasɣiːr	تصغير
Eight	θamaːnija	ثمانية	Previous	assaːbiq	السابق
Nine	tisʕa	تسعة	Next	attaːliː	التالي
Right	jamiːn	يمين	Send	ʔirsaːl	إرسال
Left	jasaːr	يسار	Receive	istiqbaːl	استقبال
Up	ʔaʕlaː	أعلى	Move	taħriːk	تحريك
Down	ʔasfal	أسفل	Rotate	tadwiːr	تدوير
Forward/Front	ʔamaːm	أمام	Record	tasʒiːl	تسجيل
Backward/Back	xalf	خلف	Enter	ʔidxaːl	إدخال
Yes	naʕam	نعم	Digit	ɾaqm	رقم
No	laː	لا	Direction	ittiʒaːh	اتجاه
Start	ibdaʔ	ابدأ	Options	xijaːraːt	خيارات
Stop	tawaqqaf	توقف	Undo	taraːʒuʕ	تراجُع

Table 1: The 40 chosen keywords with their translations and pronunciations in IPA.

precisely as follows: each contributor is shown a set of 40 slides, each slide containing a single keyword. He/She consecutively pronounces the keyword and then moves to the next slide and so on. We found that this procedure is more suitable as we implicitly had a variety of utterances and pitches because we noticed a change in their pitch after each round as they felt tired and bored gradually. After each round, the recorded

audio file is saved. This process is repeated 10 times by each contributor, and the average time we spent on collecting the data from each one on all 10 rounds was about half an hour.

The set of contributors consists of 30 people (19 males and 11 females) with various ages (ranging from 13 to 91 years old) and speaking in slightly different accents as they come from different Syrian cities. The pronunciations are performed in Standard Arabic (الفصحى) as opposed to the spoken dialect of Arabic that is used generally in Syria. We met each contributor personally for their recordings and got their consent for this publication. The recordings happened in quiet places or closed rooms to avoid recording possible background noises or private personal conversations. However, there must be some natural background noises that are difficult to be excluded unless we use a soundproof studio, which is costly.

All audio files were recorded using the same laptop. We used an external (cheap) earphone microphone most of the time, but sometimes we used the laptop's built-in microphone. The collection of these recording happened in November and December of 2019.

Each resulting audio file is a long recording containing 40 utterances (the chosen keywords) with appropriate pauses in between. Next, we need to split them into smaller audio files with each being one-second long and comprising only one keyword positioned approximately in the middle of the audio segment. We considered free Voice Activity Detection (VAD) tools, but they do not seem to give reliable results. We tested the tool that was used to extract the loudest section in the GSC dataset, but we found that some of the resulting audio files only contain background noise, and the tool sometimes clips an audio segment incorrectly due to the presence of noise at the beginning or the end, causing the loss of a significant part of the utterance. We found several such troublesome files in the GSC dataset. See the footnotes for examples of the former[5] or the latter[6] case.

It was challenging to obtain clean audio recordings of one-second length, therefore we decided to clip the files manually taking advantage of Audacity[7], a free and open-source software. The time it took us to process the data we collected from a single contributor manually was about one hour. The final structure of the dataset is shown in Figure 1.

Table 2 summarizes the key properties of the dataset and compares it to the two versions of the GSC dataset. We can see that the number of utterances per class in GSC version 0.02 is about tenfold. The GSC dataset also contains a greater variety of contributors, whereas our dataset is cleaner regarding the quality of data as the segmentation of speech was done manually.

[5] `house/0f2442b6_nohash_0.wav`, `five/0c09f202_nohash_0.wav`.

[6] `house/0a396ff2_nohash_0.wav`, `five/1a0f9c63_nohash_0.wav`.

[7] `https://www.audacityteam.org/`

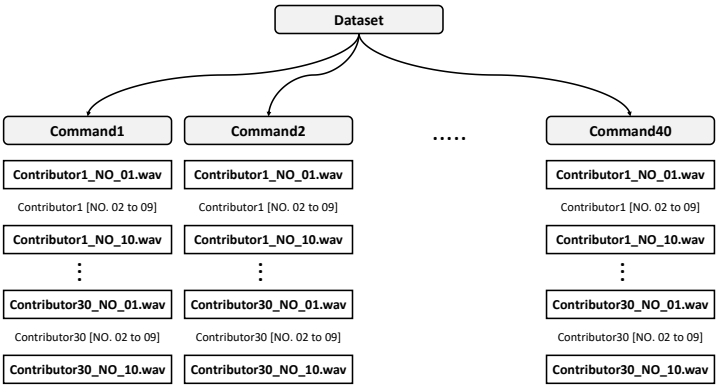

Figure 1: The final structure of the ASC dataset. There are 40 folders, each of which represents one keyword and contains 300 files. The first eight digits of each file name identify the contributor, while the last two digits identify the round number. For example, the file path `rotate/00000021_NO_06.wav` indicates that the contributor with the ID `00000021` pronounced the keyword `rotate` for the 6^{th} time.

Dataset	Date	Language	No. of Classes	No. of Files	Files per Class
ASC	2020	Arabic	40	12,000	300
GSC 0.02	2018	English	35	105,829	~3000
GSC 0.01	2017	English	30	64,727	~2160

Table 2: Comparison of the ASC dataset with the two versions of the GSC dataset.

3.4. Background Noise

KWS systems must be able to operate efficiently despite the presence of some background noise. Thus, we must train a model on utterances that contain some natural noise to be more robust in recognizing keywords in different ambient conditions. To achieve that, we also recorded background noise from different natural sources such as the noise of cars in the street (recorded inside an apartment near a window) and the noise of preparing food in a kitchen. These recordings are uploaded with the dataset. In Section 4, we describe how we used these supplementary recordings in our experiments.

4. Experiments

To benchmark our newly introduced dataset, we performed multiple experiments. We used the built-in machine learning algorithms in scikit-learn[8] and used PyTorch[9] to implement the deep learning models. Our

[8] `https://scikit-learn.org/`
[9] `https://pytorch.org/`

code is fully available online[10]. Furthermore, we prepared and uploaded Google Colaboratory notebooks to make our experiments reproducible effortlessly (see `colab` folder in the repository). We divided our experiments into two sets. In the first one, we used some classical machine learning algorithms to get a feel of the complexity of our data and to set a preliminary baseline. For the second set of experiments, we used more advanced deep learning models with different architectures and multiple augmentations.

The task is to classify the input signal into 41 different classes: the 40 chosen keywords shown in Table 1, plus a newly introduced 'silence' label, which refers to audio signals that do not contain any speech but only background noise. To keep our classes balanced, we added 300 such 'silence' samples using the following procedure: (1) Randomly select a background noise file from those described in section 3.4 with probabilities proportional to their lengths. (2) Uniformly sample a one-second audio segment from the selected file. (3) Multiply its amplitude by a factor uniformly sampled from $[0, 0.5)$ as it was necessary to reduce the amplitude of the loud noise, while the randomness helps to obtain a variety of noise levels in the silence samples.

As a performance metric, we simply used the accuracy value, i.e., the portion of correctly classified samples divided by the total number of samples. This is simply motivated by the fact that our data is balanced and the accuracy value was reported and used as a performance measure in related previous works. For specific applications, one might incorporate or consider the use of other metrics such as false alarm rate and false reject rate and optimize the alerting threshold based on the intended behaviour of the application.

4.1. Data Split

As commonly done in machine learning settings, we split the dataset into three subsets: training, validation, and testing. Considering the number of instances in our dataset, we decided to keep 60% of them as the training set, 20% as the validation set, and the remaining 20% are kept as a hold-out testing set. For each subset, we created a Comma-Separated Values (CSV) file containing the relative path of each audio file with its corresponding label.

In our split method, we guarantee that all recordings of a certain contributor are within the same subset. In this way, we avoid having signals with some similarities in both the training and validation/testing sets, as this may affect the validity of the results. Besides, it makes sure that the model will learn to generalize to new people outside of our dataset.

Please note that the ordering of contributors is meaningless. However, consecutive contributors probably did their recordings in the same environment (e.g., in the same room/house). Therefore, we created a random permutation of contributors' IDs before distributing their recordings over the different subsets to avoid having data of consecutive contributors in the same subset. Experimentally, not doing so caused a big difference between training, validation, and testing results.

[10] `https://github.com/fresher96/arabic-speech-commands/`

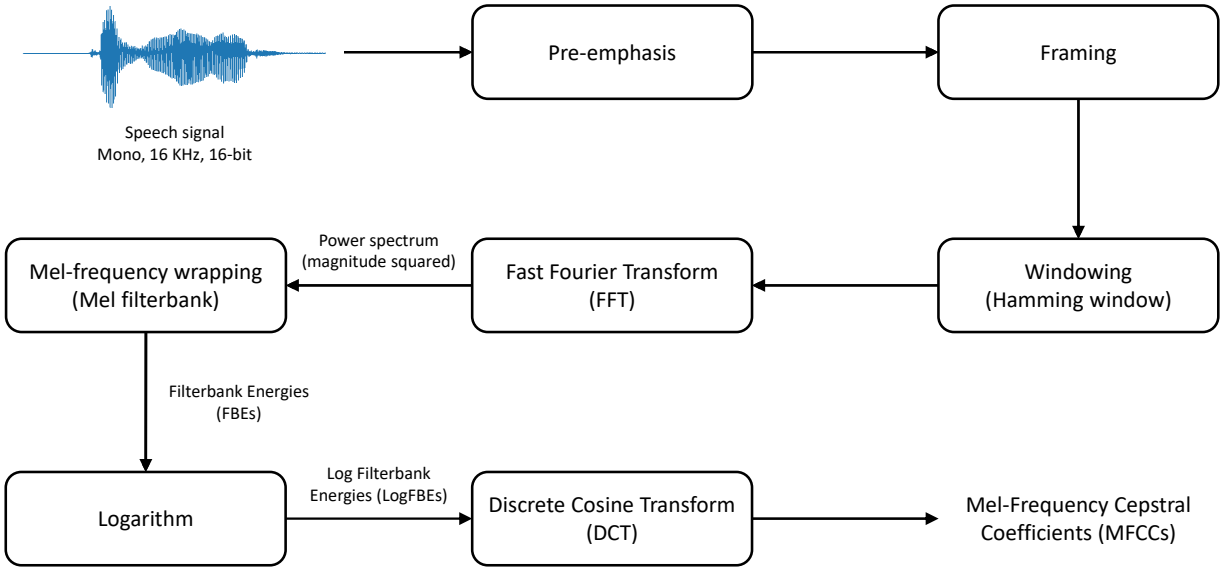

Figure 2: The main steps of the MFCCs feature extraction process.

4.2. Data Preprocessing

MFCCs (Huang et al., 2001, Sec. 6.5.2) are one of the most widely used features to represent speech signals in ASR systems. Although it is not the only one, it is known to help achieve remarkable results compared to other features (Davis and Mermelstein, 1980), and this prompted us to use it in our experiments. The main steps in the MFCCs feature extraction process are illustrated in Figure 2. Specifically, in the framing step, we used a window of length 25 ms with a step length of 10 ms. We also used 40 Mel-filters to obtain the Log Filterbank Energies (LogFBEs) and applied the Discrete Cosine Transform (DCT) on these values to extract 13 MFCC features. Then, we discarded the first feature, as it does not carry useful information, and took only the next 12 ones. Note that we used PyTorch implementation for this process.

Figure 3 shows a comparison of the output of different steps in the preprocessing pipeline, namely: Spectrograms, LogFBEs, and MFCCs. We can notice how the visualization of LogFBEs is very similar to the spectrogram, as it clearly shows how the signal's energy is distributed over a range of Mel-frequencies, and how the shape of the spectrum changes over time. While in MFCCs, the most significant values exist in the first few coefficients.

4.3. Preliminary Results

Table 3 shows the results of the first set of experiments we applied to the dataset. These experiments did not include augmentations and were done with minimal hyperparameters tuning on multiple classical

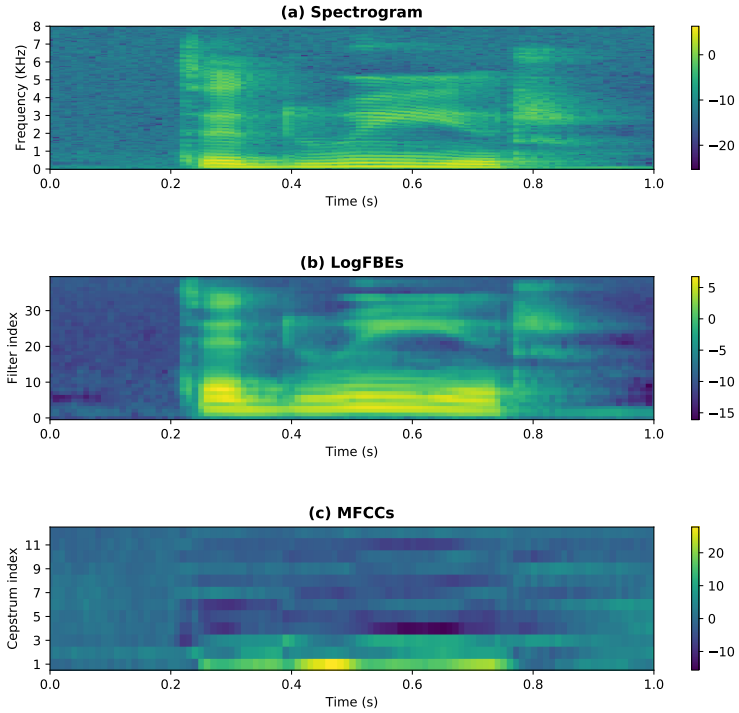

Figure 3: A comparison between Spectrograms, LogFBEs, and MFCCs using an instance from our dataset.

machine learning algorithms to get an initial baseline. Our goal here is to gauge the complexity of this classification task on the ASC dataset. With that in mind, we used a variety of classifiers that incorporate different assumptions and learn using a variety of procedures. It is worth to mention that all of these classifiers do multi-class classification because we tried to use Support Vector Machines (SVMs) as binary classifiers but their training would be infeasible as we would need to use one-versus-one classification and train 820 ($= (41 \times 40)/2$) classifiers.

The tuning procedure was performed manually. We first tried to identify the most important hyper-parameters that are sensitive and correspond to big changes in the resulting accuracy on the validation dataset. Then we defined a range of possible values for each one of these hyperparameters and used grid search. Finally, we arrived at the following values: for K-Nearest Neighbors (KNN), we used 6 neighbors and a weight equal to the inverse of the Euclidean distance; in Naive Bayes, there are no hyperparameters; the Decision Tree was limited to a max depth of 700 and we allowed 700 features to be considered when looking for the best split; in the Random Forest we used 250 estimators each with a max depth of 325, and we allowed 10 features when looking for the best split; in Gradient Boosting Trees (GBT) we used 175 estimators each with a max depth of 5, we allowed 6 features when looking for the best split, and the learning rate was set to 0.1; Logistic Regression penalty parameter for regularization was set to 10^{-5}; Finally, for the Hidden Markov Models (HMMs), we built one model for every class, each model had 26 hidden states

and used Gaussian emission probabilities. For references on these methods, see (Rabiner, 1989) for HMMs, and see (Hastie et al., 2001; Géron, 2017) for the others.

Table 3 also shows NN-based approaches. We tried three different models outlined below. All of these models use: cross-entropy loss with a softmax output layer of 41 units (the number of classes); adam optimizer (Kingma and Ba, 2014) with 10^{-3} initial learning rate; Rectified Linear Unit (ReLU) activation function in the hidden layers; 32 mini-batch size and 10^{-3} L2 weight decay; and a learning rate scheduler that multiplies the learning rate by 0.1 when the training loss plateaus for 5 consecutive epochs. These models are trained for 50 epochs.

4.3.1. DNN

A simple feedforward fully connected neural network (Goodfellow et al., 2016, Ch. 6) of 3 stacked hidden layers of 256 hidden units in each. Each layer is succeeded by a batch normalization layer (Ioffe and Szegedy, 2015) (note that Logistic Regression can be viewed as a simple case of DNNs).

4.3.2. CNN

A convolutional neural network (Goodfellow et al., 2016, Ch. 9) of 4 stacked convolutional layers with 16, 32, 64, and 128 channels (feature maps), respectively. Each layer is followed by batch normalization (Ioffe and Szegedy, 2015) and a 2×2 max-pooling layer. Finally, these layers are succeeded by a dropout layer (Srivastava et al., 2014) with 0.25 omission probability and a fully connected feedforward layer with 256 hidden units.

4.3.3. LSTM

A recurrent neural network (Goodfellow et al., 2016, Ch. 10) with two LSTM layers (Hochreiter and Schmidhuber, 1997), followed by a dropout layer (Srivastava et al., 2014) with 0.5 omission probability, and finally succeeded by two Time Distributed fully connected layers with 128 and 64 hidden units.

4.4. Augmentations

To give a better benchmark, we used augmentations with the NN-based approaches. We performed two different types of augmentations: augmentations that apply to the raw waveform signal, and augmentations that apply to the 2D-shaped extracted features. For an illustration of these augmentations see Figure 4. The final pipeline is as follows: (1) Raw waveform signal as 1D input. (2) Time domain augmentations (Time Shifting + Background Noise Adding). (3) Feature extraction (the output of this step is 2D). (4) Spectral/Cepstral augmentations (Time Masking + Frequency Masking). (5) Deep learning model. (6) The predicted keyword as output.

Model	Val	Test
KNN	71.50	67.72
Naive Bayes	74.55	71.18
Decision Tree	38.01	35.16
Random Forest	77.07	75.49
GBT	71.30	72.40
Logistic Regression	74.39	**75.77**
HMM	63.62	60.85
DNN	81.02	81.91
CNN	89.96	**91.10**
LSTM	83.37	83.54

Table 3: The Accuracy on the validation and testing subsets obtained from applying different machine learning and deep learning models to the ASC dataset without using any augmentations.

4.4.1. Time Shifting

Shift the signal with an offset uniformly sampled from $[-0.2, 0.2)$, then pad it with zeros to keep its one-second length.

4.4.2. Background Noise Adding

We sampled background noise the same way as we sampled the 'silence' instances, then we added this noise to the input audio signal. This augmentation helps the model overcome the presence of background noise in the utterances and to generalize to different environments.

4.4.3. Time Masking

Mask out a randomly chosen band in the time axis/dimension (of the 2D array of features) with zeros (Park et al., 2019). We used two such augmentations with the mask width randomly sampled from $[[0, 8]]$. This augmentation helps train the model to overcome discontinuities in time.

4.4.4. Frequency Masking

Mask out a randomly chosen band in the frequency/cepstral axis/dimension (of the 2D array of features) with zeros (Park et al., 2019). The mask width is randomly sampled from $[[0, 3]]$. This augmentation helps train the model to overcome a partial loss of frequency information.

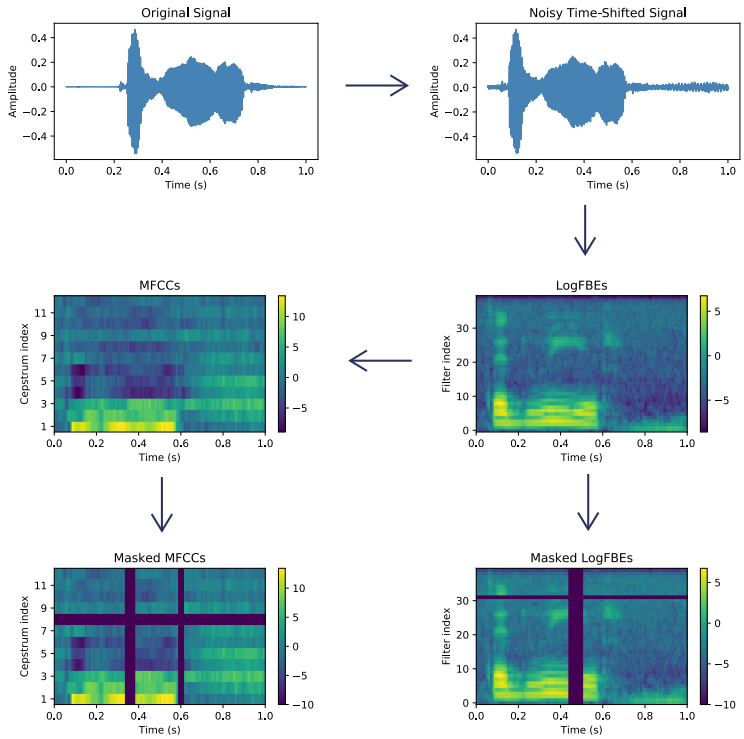

Figure 4: Illustration of the augmentations we applied. The first two augmentations (time shifting + noise adding) apply before feature extraction. The second two augmentations mask the extracted 2D array of features on both axes/dimensions. Please note that the masked region in this figure was set to a small value instead of zero for illustration purposes only.

5. Results & Discussion

Table 4 shows the final results of applying the three architectures in Section 4.3 with the augmentations in Section 4.4. In the first three rows, we used the same settings as before but trained the models for 75 epochs instead of 50. We can see how these simple augmentations helped greatly in improving the accuracy of our models on the validation/test set.

In the last three rows, we used LogFBEs as inputs instead, i.e., we skipped the last step of applying the DCT (see Figure 2). We can see that this did not help much, although now we have 40 features for each frame instead of 12. This proves that DCT is effective in our case and can be used as a feature selection/compression preprocessing step. However, note that skipping the DCT will increase the number of multiplications and parameters in the model.

Figure 5 shows the learning curves of the best performing model (the CNN-MFCC model corresponding to the second row in Table 4).

In comparison with state-of-the-art (SotA) models that use the GSC dataset, we see that our results are slightly better (SotA accuracy is around 97.5% on both GSC versions (Majumdar and Ginsburg, 2020)). This comparison is interesting given that GSC was published more than two years ago, and that recent

Model	Features	Val	Test	Par.
DNN	MFCCs	90.16	86.83	454,185
CNN	MFCCs	97.89	**97.97**	**305,033**
LSTM	MFCCs	96.26	95.85	494,633
DNN	LogFBEs	91.91	90.65	1,178,153
CNN	LogFBEs	97.52	**97.93**	**501,641**
LSTM	LogFBEs	95.57	94.96	508,969

Table 4: The Accuracy of three different architectures applied on MFCCs and LogFBEs. The last column is the number of learnable parameters in each model.

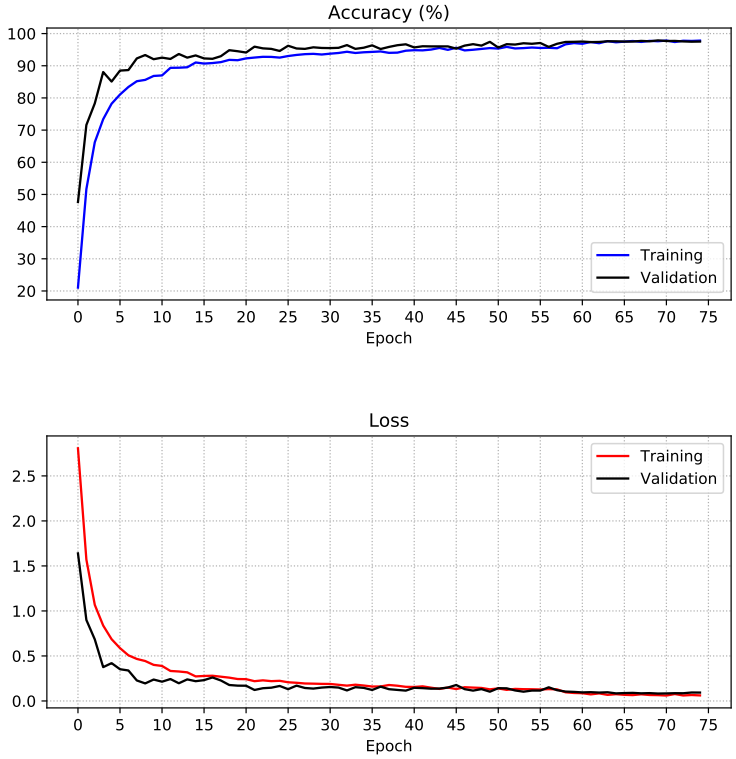

Figure 5: The learning curves of the best performing model (CNN-MFCC row in Table 4).

approaches use much more complicated architectures and techniques. Besides, ASC has 5 to 9 times fewer samples than GSC, and 5 to 10 more classes (see Table 2).

5.1. Streaming Audio Tests

We wrote a naive program to test the best performing model (CNN-MFCC with augmentations) using streaming audio from a microphone. Although this program could be improved (e.g., by smoothing the posterior probabilities as in (Chen et al., 2014; Sainath and Parada, 2015)), we just wanted to make sure that it can recognize the set of predefined keywords in an audio stream correctly, and therefore these models can be adjusted or trained explicitly for such task.

Two persons that were not included in the dataset did this testing. Each of them performed two rounds using two different microphones in two different environments. As a result, we had a total of 154 correct predictions out of 160 utterances resulting in 96.25% accuracy.

In general, when testing a model on a stream of audio, a new type of utterances should be handled; utterances that do not belong to the set of keywords of interest. Related works (Chen et al., 2014; Sainath and Parada, 2015; Warden, 2018; Tang and Lin, 2018) introduced a new class called 'unknown' or 'filler' and explicitly trained their models to handle these negative samples as a separate class, i.e., they handled it in the same manner as we handled 'silence' in our approach. For example, (Warden, 2018; Tang and Lin, 2018) chose 10 classes from the GSC dataset as their keywords of interest, while they used the rest as 'unknown' utterances, and also added the 'silence' instances. However, due to the specific characteristics of this class, we believe that this handling is inadequate because the model will be trained explicitly to reject negative utterances present in the dataset, i.e., it will not be able to reject new utterances or words that were not presented in the dataset as it did not hear them before. For this, we suggest training an auxiliary model as an anomaly detector to explicitly capture the probability distribution of the predefined keywords and reject other audios that do not fit in this distribution.

6. Conclusion & Future Work

In this paper, we reviewed the importance of KWS tasks and how they differ from other ASR tasks. We introduced ASC, our newly built dataset that was inspired by the GSC dataset, and elaborated on its collection procedure, highlighting the similarities and differences between the two datasets. Lastly, we reported the experiments that we conducted to benchmark our dataset. We performed classical approaches to make sense of the data and its complexity. Then, we performed more advanced deep learning based approaches with data augmentations, which resulted in near-perfect accuracy.

With the introduction of ASC for KWS in Arabic, now it is possible to combine it with GSC to have an even larger dataset that can serve as a multilingual dataset for KWS tasks. For future work, it is worth investigating the performance that can be achieved on a stream of speech more systematically.

Acknowledgement

We are very grateful to the 30 contributors that donated their recordings to this dataset. We also thank Imad Eddine Ibrahim Bekkouch for his helpful comments.

References

de Andrade, D.C., Leo, S., Viana, M.L.D.S., Bernkopf, C., 2018. A neural attention model for speech command recognition. arXiv preprint arXiv:1808.08929 .

Benamer, L., Alkishriwo, O., 2020. Database for arabic speech commands recognition, in: Proceedings of the 3rd Conference for Engineering Sciences and Technology. URL: http://dspace.elmergib.edu.ly/xmlui/handle/123456789/195.

Bengio, Y., De Mori, R., Flammia, G., Kompe, R., 1992. Global optimization of a neural network-hidden markov model hybrid. IEEE transactions on Neural Networks 3, 252–259.

Chen, G., Parada, C., Heigold, G., 2014. Small-footprint keyword spotting using deep neural networks, in: 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE. pp. 4087–4091.

Davis, S., Mermelstein, P., 1980. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. IEEE transactions on acoustics, speech, and signal processing 28, 357–366.

Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L., 2009. Imagenet: A large-scale hierarchical image database, in: 2009 IEEE conference on computer vision and pattern recognition, IEEE. pp. 248–255.

Géron, A., 2017. Hands-on machine learning with Scikit-Learn and TensorFlow. O'Reilly Media, Sebastopol, CA.

Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep Learning. MIT Press. http://www.deeplearningbook.org.

Graves, A., 2012. Hierarchical Subsampling Networks. Springer Berlin Heidelberg, Berlin, Heidelberg. pp. 109–131. URL: https://doi.org/10.1007/978-3-642-24797-2_9, doi:10.1007/978-3-642-24797-2_9.

Graves, A., Jaitly, N., 2014. Towards end-to-end speech recognition with recurrent neural networks, in: Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32, JMLR.org. p. II–1764–II–1772.

Hastie, T., Tibshirani, R., Friedman, J., 2001. The Elements of Statistical Learning. Springer Series in Statistics, Springer New York Inc., New York, NY, USA.

Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural Comput. 9, 1735–1780. URL: https://doi.org/10.1162/neco.1997.9.8.1735, doi:10.1162/neco.1997.9.8.1735.

Huang, X., Acero, A., Hon, H., 2001. Spoken Language Processing: A Guide to Theory, Algorithm, and System Development. Prentice Hall PTR. URL: https://books.google.ru/books?id=reZQAAAAMAAJ.

Ioffe, S., Szegedy, C., 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167 .

Kingma, D.P., Ba, J., 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 .

Majumdar, S., Ginsburg, B., 2020. Matchboxnet–1d time-channel separable convolutional neural network architecture for speech commands recognition. arXiv preprint arXiv:2004.08531 .

McMahan, B., Rao, D., 2017. Listening to the world improves speech command recognition. arXiv preprint arXiv:1710.08377 .

Park, D.S., Chan, W., Zhang, Y., Chiu, C.C., Zoph, B., Cubuk, E.D., Le, Q.V., 2019. Specaugment: A simple data augmentation method for automatic speech recognition. Interspeech 2019 URL: http://dx.doi.org/10.21437/Interspeech.2019-2680, doi:10.21437/interspeech.2019-2680.

Rabiner, L.R., 1989. A tutorial on hidden markov models and selected applications in speech recognition. Proceedings of the IEEE 77, 257–286.

Sainath, T.N., Parada, C., 2015. Convolutional neural networks for small-footprint keyword spotting, in: Sixteenth Annual Conference of the International Speech Communication Association.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R., 2014. Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research 15, 1929–1958.

Tang, R., Lin, J., 2018. Deep residual learning for small-footprint keyword spotting, in: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE. pp. 5484–5488.

Wang, D., Lv, S., Wang, X., Lin, X., 2018. Gated convolutional lstm for speech commands recognition, in: International Conference on Computational Science, Springer. pp. 669–681.

Warden, P., 2018. Speech commands: A dataset for limited-vocabulary speech recognition. arXiv preprint arXiv:1804.03209 .