EFFICIENTWORD-NET: AN OPEN SOURCE HOTWORD DETECTION ENGINE BASED ON ONE-SHOT LEARNING

Chidhambararajan R, Aman Rangapur, Dr. Sibi Chakkravarthy

Vellore Institute of Technology-AP, India

ABSTRACT

Voice assistants like Siri, Google Assistant, Alexa etc. are used widely across the globe for home automation, these require the use of special phrases also known as hotwords to wake it up and perform an action like "Hey Alexa!", "Ok, Google!", "Hey Siri!" etc. These hotwords are detected with lightweight real-time engines whose purpose is to detect the hotwords uttered by the user. This paper presents the design and implementation of a lightweight, easy-to-implement hotword detection engine based on one-shot learning which detects the hotword uttered by the user in real-time with just one or few training samples of the hotword. This approach is efficient compared to existing implementations because the process of adding a new hotword in the existing systems requires enormous amounts of positive and negative training samples and the model needs to retrain for every hotword. This makes the existing implementations inefficient in terms of computation and cost. The architecture proposed in this paper has achieved an accuracy of 94.51%.

1. INTRODUCTION

With the advent of the Internet of Things (IoT) and home automation, there is a growing need for voice automation in edge devices, but running a heavy Text To Speech (TTS) Engine is too computationally expensive in these edge devices [1]. Instead, we can run engines that need to listen for specific activation phrases called "Hotwords" to perform certain actions since the detection of hotwords is computationally less expensive than full-blown TTS engines [28].

Lightweight models [28] are trained for detecting these hotwords from audio streams. This is used to save resources from heavy models such as speech recognition from running all day. The Core application for hotword detection is shown in Fig. 1.

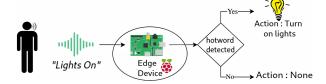Convolutional Neural Networks (CNNs) have proven to

Figure 1: Overview of the proposed model.

be the best in analysing image data [6]. Audio files are converted into Log Mel spectrograms where various frequencies are distributed on the Mel scale and plotted as an image [2, 18]. This image data is further analysed by the CNNs to get maximum optimacy [6]. The base network is further attached to a Siamese network which learns to output embedding vectors with less distance for similar hotwords and huge distance for dissimilar hotwords. This way a state of the art accuracy is achieved for hotword detection with fewer audio examples of the hotwords. To our knowledge, this is the first attempt to solve the problem of retraining hotwords with one shot / few-shot learning. This approach is highly inspired by Face-Net [19], one-shot learning deployed for face recognition, which allows us to add a new face to the system without retraining the model [19]. Except for the well-known hotword detection engines with low accuracy, other engines require huge datasets with positive and negative samples for training new hotword and all of these are closed source [28, 9].

EfficientNet [20] is one of the most efficient CNN architecture to the date, and the first four blocks of EfficientNet (B0 variant) is chosen for the base model in the Siamese network. To train the Siamese network, positive and negative pairs of audios are given to the network and trained to output 1, 0 for positive pairs and negative pairs respectively to know how similar the pronunciations of words. In the edge device, raw audio is continuously read(with 1 sec time window), converted to Log Mel spectrogram, from which real-time vector embeddings are calculated, these embeddings are compared against a pre-calculated vector embedding of

the desired hotword for similarity [27].

2. EXISTING RELATED WORKS

The problem of detecting hotwords from audio streams started ever since the advent of voice-enabled IoT devices [15, 14, 17, 24, 25, 5, 29, 13, 3, 26, 22]. Porcupine [16] is a closed source hotword detection framework which detects hotwords but requires a commercial licence. It has an accuracy of 94%. A customizable hotword detection engine called Snowboy can be used to create your own hotwords [28]. It is a closed source project and requires a huge amount of data samples. PocketSphinx [9] is a lightweight variation of the CMU-Sphinx [9], an offline Speech-to-Text (STT) engine with low accuracy.

There are other frameworks such as howl [21], these frameworks require large amounts of positive and negative samples of the hotword to train and recognize the new words. Snowboy [4] and howl's [21] accuracy depends on the size of training dataset for each hotword. STT engines can also be used to detect hotwords [1], existing engines STT [12] have very high accuracy but require a constant internet connection and an expensive subscription to their cloud service to run 24/7. Offline STT open source engines like DeepSpeech (from Mozilla) [1] and Silero [23] have good accuracy, yet require lots of on-device resources, hence cannot be run 24/7. Rhino [11], an on-device STT engine achieved the best among existing engines, by giving better accuracy and low resource requirement but closed source and requires a commercial license.

Most existing audio processing neural networks employ the usage of Log Mel Spectrograms [2] and Mel Frequency Cepstral Coefficient [2] since it conveys a better picture of the audio than conventional audio stream bits, this is due to the representation of changes for various frequencies across the audio streams. The initial development of hotword detection was achieved without noise [7] and then deployed with active noise cancellation in real-time [8]. Active noise cancellation is often achieved with hardware first or software first or hybrid approaches. The software first approaches require audio samples recorded from the ambient space. Out of these recorded sample metrics, the maximum amplitude for noise is calculated and used as a threshold for voice activity detection. Acoustic noise reduction is applied over real-time audio with the help of obtained noise only audio samples. This approach is not practical since noise only audio samples recorded from ambient space are required. To circumvent this issue, edge voice assistant's like Google, Amazon's Alexa often deploy multi-microphone array systems [8] to gather audio from all directions and separate speech audio with ease. This is a relatively simpler task as devices are surrounded with uniform noise in all directions but speech audio is not uniform in all directions. This al-

lows speech audio to be separated from noise audio. The separated speech audio is of high quality thereby helping in achieving low False Acceptance Rates (FAR).

In a hybrid approach, hardware-based noise cancellation is utilized and the audio processing neural networks are often trained with noise, this allows the system to achieve exceptionally low FAR [8].

Existing audio processing neural networks are designed as audio classification neural networks [6]. The network needs a lot of initial layers to understand the audio fragment. Later, half a dozen of layers are required for the network to understand the logic of classification. These extra layers dedicated for classification requires additional computational time thereby crippling the model while running it on real-time edge devices. In this paper, this problem was resolved using EfficientNetB0 as a base network with one-shot learning. In one-shot learning, the network requires a lot of layers to understand audio samples, but the need for additional layers to understand classification is waived off, therefore, resulting in faster inference in the edge devices.

As mentioned, the existing lightweight models [28, 9] are binary classifiers that need to be retrained with a huge number of negative and positive samples for a new hotword, this results in expensive and inefficient gathering of datasets [9]. These networks needs to be retrained for newer hotwords again. Also, these engines are closed source where there is no scope of development in the future and users need to spend lots of money for the useage. Hardware first and a hybrid approach are applicable in the scenarios where the edge device's hardware specifications are under the control of developers like edge voice assistants like Alexa [7]. But this approach doesn't work well when the edge device is not designed by developers [8]. Hence, there is a need for audio processing neural networks to be trained with very high amounts of noise to work robustly without need of hardware interventions.

EfficientWord-Net is an open source engine that solves the process of retraining the model for new hotwords by eliminating the requirement of huge datasets. It works efficiently with the audio samples with decent noise added in the background with a great inference time on small devices like Raspberry Pi. Moreover, our system outperformed previous existing approaches in terms of accuracy and inference.

3. ONE-SHOT LEARNING/SIMILARITY LEARNING

One of the demanding situations of face recognition/hotword recognition is to achieve performance with fewer samples of the target, which means, for maximum face recognition programs, model should recognize a person given with the aid of using one photo of the man or
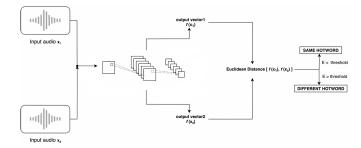
Figure 2: One-shot learning architecture for hotword detection

woman's face. Traditionally deep learning algorithms do not work well with the simplest one training example or one data point for a class. In one-shot learning, a model learns from one sample to apprehend or recognize the person, and the industry needs most face recognition models to use this due to the fact a company has one or few images of every of their personnel in the database.

Similarity learning is a type of supervised learning where a network is trained to identify the similarity between 2 data points of the same class instead of classification or regression. The network is also trained to learn the dissimilarity between 2 data points of different sources. This similarity is used to determine whether an unlabelled data point belongs to the same class or not.

When 2 images are fed to a neural network to learn the similarity between them (inputs two images and outputs the degree of difference between the two images), the output would be a small number if the two images are of the same person. And if they are of two different people, the output would be a large number. However, the use of different types of functions keeps the value between 0 and 1. A hyper parameter(τ) is used as a threshold if the degree of difference is less than the threshold value, these pictures belong to the same person and vice versa. Similarly, in this paper, a threshold value of 0.2 is defined, if the degree of difference between the two hotwords is less than 0.2, those two hotwords are the same, else they are different.

d(*hotword1, hotword2*) = degree of difference between hotwords.
d(*hotword1, hotword2*) $< \tau$, both hotwords are same.
d(*hotword1, hotword2*) $> \tau$, hotwords are different.

This approach of feeding 2 different images to the same convolutional network and comparing the encodings of them is called a Siamese Network [27].

In Figure. 2, two raw audio segments are fed to the same neural network, output encodings of input audios are captured, Euclidean distance between these two vectors are calculated, if the Euclidean distance is less than the threshold value, the hotwords in the audio are same and vice versa.

4. PROPOSED METHOD

4.1 Preparation Of Dataset

The dataset used for training the network is homegrown artificially synthesized data that was made with naturally sounding neural voices from Azure Cloud Platform and Siri. Furthermore, the voices were selected to include all available voice accents with the respective countries to ensure better performance across accents and gender. Hotword detection does not rely on a word's meaning but only on its pronunciation. So, to generate the audio, a pool of words is selected in which each word sounds unique compared to the other words in the pool. Google's 10000-word list [10] is used while training the model, and these words were converted to respective phoneme sequences. The sequences were checked for similarity, the words which shared $\geq 80\%$ of the same phoneme sequence were removed to ensure low similarity in pronunciation among the word pool. The word pool was converted to the audio pool with the above mentioned text-to-speech services. The generated audio is combined with noises such as traffic sounds, market place sounds, office and room ambient sounds to simulate a naturally collected dataset. For each word, 5 such audio samples were generated in which each sample had a randomly chosen voice and randomly chosen background noise to ensure variety. The randomly chosen noises are then combined with the original audio with a noise factor randomly chosen between 0.05 to 0.2 (noise factor is a fraction of noise's volume in the resulting audio). A sampling rate of 16000 Hz was chosen since audio quality below 16000 Hz became very poor. Finally, the audios are converted to Log Mel Spectrograms.

Two audios generated from the same word are chosen to make the true pair and two audios generated from two different words are chosen to make the false pair. The total number of true and false pairs generated in this method were 2694. 80% of the data was split for training, and remaining 20% was used as testing data to eliminate over-fitting. The network is fed with the true pair and false pairs to output a higher similarity score for true pairs and lower similarity score for false pairs.

4.2 Network Architecture

The input for each base network is 98x64x1 (Refer Fig. 3. The base network of the model is made up of the first four blocks of EfficientNetB0 architecture [20]. The output from the EfficientNet layers is processed further with Conv2D layer with 32 filters and 3 stride values, processed by batch normalization and max pool layer, this is fed to a similar

Figure 3: Base network Block.

Table 1: Euclidean Distance and Similarity score

Euclidean Distance	Similarity Score
0	1.0
$< \tau$	1.0 - 0.5
τ	0.5
$> \tau$	0.5 - 0

stack of layers. This is done to reduce the number of feature points efficiently. Finally, the output from the convolutions is flattened and attached to the dense layer with 256 units followed by L2 regularization to give the reduced vector approximation of the input.

A true pair or a false pair in the dataset is fed to two parallel blocks of the base network, where these parallel blocks share the weights. Euclidean distance between corresponding output vector pairs is calculated. Table. 1 shows the Euclidean distance mapped to similarity score to the scale 0-1.0.
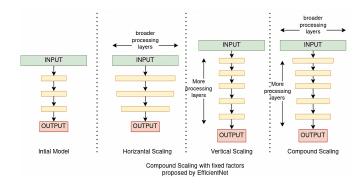

Figure 4: Model Scaling in EfficientNet.

4.3 Training Parameters and Loss Function

The problem of analysing raw audio to examine images is resolved by converting the audio into Log Mel spectrograms images, this helps the neural network by allowing it to directly analyse frequency distribution over time, learning first to identify different frequencies and analyse them.

These generated images were fed to a convolution neural network that follows the EfficientNet Architecture (Refer Fig. 4). It is a convolutional neural network architecture and a scaling method that uniformly scales all dimensions of depth/width/resolution using a compound coefficient.

This network architecture was chosen since the accuracy was similar to that of ResNet which held the previous state of the art top5 accuracy of 94.51%. Moreover, the number of parameters in ResNet was 4.9x higher than EfficientNetB0's parameter count, thereby making it computationally more efficient than ResNet. Only 4 blocks of EfficientNetB0 were taken for the base network, the output was further attached to a Conv2D block, which was later flattened and l2 normalized to give the output vector.

Conventional Siamese neural networks use triplet loss where a baseline (anchor) input is compared to a positive(true) input and a negative(false) input with vector distance calculation metrics such as Euclidean distance, cosine distance, etc.

$$\text{Triplet Loss Function} = \max\left(\|f(x^a) - f(x^p)\|^2 - \|f(x^a) - f(x^n)\|^2, 0 \right)$$

x^a is an *anchor* example.
x^p is a *positive* example that has the same identity as the anchor.
x^n is a *negative* example that represents a different entity.

The triplet loss function makes the neural network minimize the distance from the baseline(anchor) input to the positive (true pair), and maximize the distance from the baseline(anchor) input to the negative (false pair). It is also equipped with a threshold, which forces the network to reduce the distance between true pairs below the threshold and false pairs beyond the threshold.

The calculated distance between the pair is sent through a function F(x) which outputs close to 1 when distance is low and 0 when distance is high, thereby making the function give a score close to 1 for similar pairs and close to 0 for dissimilar pairs. This was done so that the network will be able to tell similarities between a pair of samples in terms of percentage.

$$F(x) = 1 - \frac{x^4}{(\tau^4 + x^4)}$$

4

Here x is the calculated distance between the vectors, this function gives 1 when x is 0, gradually reduces to 0.5 when x = τ (threshold,t[in the above equation]) and eventually to zero. The function is symmetric making f(x) go to zero when x<0, with Euclidean distances $\geq$ 0.

For each true pair, the ground truth was set 1 and for each false pair the ground truth was set to 0, this allowed us to treat the problem as a binary classification problem and easily apply binary cross-entropy loss function while training the engine.

Binary cross-entropy loss is defined by

$$loss = -\frac{1}{N} \sum_{i=1}^{N} y_i \bullet log\left(p\left(y_i\right)\right) + \left(1 - y_i\right) \bullet log\left(1 - p\left(y_i\right)\right)$$

4.4 Optimization Strategies

4.4.1 Log Mel Spectrogram:

Many audio processing neural networks directly process the audio with several Conv1D layers stacked on top of each other to make the network understand the audio and it has a drawback. The network would first have to understand the concept of frequency and check for the distribution of various frequencies across the audio. This forces the network to allocate its initial Conv1D layers to understand the concept of frequency and learn to check for the distribution of various frequencies across the audio, which would be further analysed by the next layers to make sense of the audio. These additional preprocessing layers can be skipped by going for a Log Mel Spectrogram (a heatmap distributing various frequencies across the audio).

Since the network is being directly fed with the distribution of various frequencies across the audio, the network can allocate all of its resources to make sense of the audio directly, thereby enhancing the accuracy of the system. This method is largely inspired by Google's TensorFlow magenta (a set of audio processing tools for TensorFlow).

4.4.2 L2 Regularization to an output vector of base network:

Initially, the network was trained with no L2 regularization, the accuracy was never able to cross 89.9%, we have tried various techniques out of which adding L2 regularization to the output vector of the base network gave the best performance. With that added, our network was able to reach 95% accuracy with noise. This performance increase was due to the fact that L2 regularization confined the output vector to the surface of a 256-dimensional unit sphere at the origin; without this confinement, it was very easy for the network to give huge distances for false pairs. Hence, the network eventually became biased, worked well only for false pairs
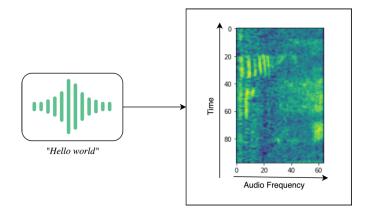

Figure 5: Sample audio converted to Log Mel Spectrogram.

and didn't give small distances for true pairs resulting in the performance drop.
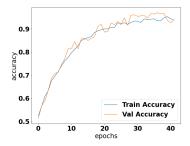
5. RESULTS

The network was trained with noise using a batch size of 64, 75 training steps per epoch for 42 epochs beyond which the training loss started to oscillate. Adam optimization algorithm was used with an initial learning rate of 1e-3, this learning rate was reduced by a learning rate scheduler which checks for lack of decrease in training loss for 3 epochs and reduces the learning rate by a factor of 0.1. The reduction of the learning rate was stopped when the learning rate reached an absolute minimum of 1e-5. Early stopping was scheduled with patience of 6 which stops the training process when the training loss starts to oscillate in more than 6 epochs resulting in the maximum validation accuracy of 94.51%. Fig. 6a shows the training graph accuracy/loss vs epochs with noise.
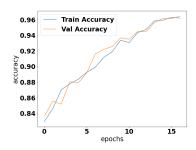
The maximum validation accuracy reached without noise is 96.8%, this accuracy was achieved by retraining the previous model with the same hyperparameters and noise factor = 0 for 14 epochs. Fig. 6a - Fig. 6d shows the training graph for epoch/accuracy vs loss with and without noise. Since the real-time audio is chunked into 1-sec windows with 0.25-sec hop length for inference, the base network was found to perform inference in 0.08 seconds in Raspberry Pi 4. Hence, the model will be able to perform inference from real-time audio streams in edge devices with no latency issues. Fig. 6b shows the noise training graph for loss vs epochs.

After performing significance test, the resulting trained model was benchmarked with other hotword detection systems on Raspberry Pi 3 clocked at 1.2GHz (4 core) and displayed in the Table. 2 and was found to outperform existing closed source models in terms of accuracy by a small level.
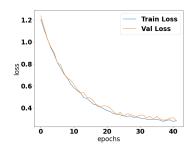
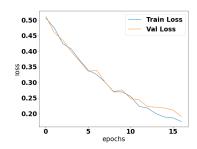For a given sensitivity value, False Rejection Rate (FRP)

(a) Model training(Epochs vs Accuracy) with noise

(b) Model training(Epochs vs Loss) with noise

(c) Model training(Epochs vs Accuracy) without noise

(d) Model training(Epochs vs Loss) without noise

Figure 6: Model Accuracy and Loss

(a) FRP vs FAR for hotword Alexa

(b) FRP vs FAR for hotword computer

(c) FRP vs FAR for hotword people
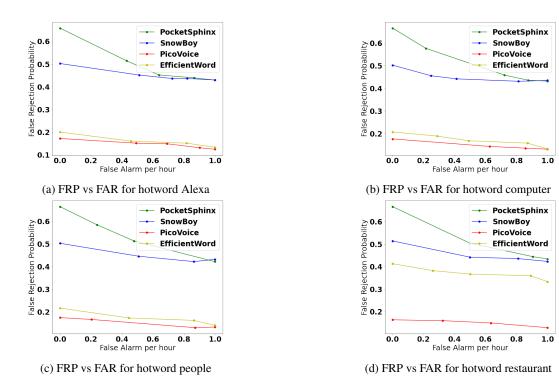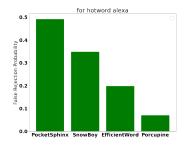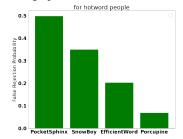

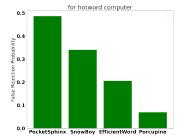(d) FRP vs FAR for hotword restaurant

Figure 7: FRP vs FAR for different Hotwords

(a) Bar graph for FRP for hotword Alexa

(b) Bar graph for FRP for hotword computer

(c) Bar graph for FRP for hotword people

(d) Bar graph for FRP for hotword restaurant

Figure 8: Bar graphs for FRP of different Hotwords

Table 2: Model benchmarks

Model	Accuracy	Inference
Efficientword-Net (Current paper)	94.51%	0.071ms
Porcupine [16]	94.78%	0.02ms
PocketSphinx [9]	54.23%	0.076ms
Snowboy [28]	88.43%	0.091ms

– (True Negatives) is measured by playing a set of sample audio files which include the utterance of the hotword, and then calculate the ratio of rejections to the total number of samples. False Acceptance Rate (FAR-False Positives) is measured by playing a background audio file which must not include any utterance of the hotword calculated by dividing the number of false acceptances by the length of the background audio in hours. Figures Fig. 7a - 7d and 8a - 8d illustrate FRP vs FAR and model performance against existing implementations for various hotwords. All these hotwords are not included in the training dataset.

6. CONCLUSION

In this paper, we proposed a one-shot learning-based hotword detection engine to solve the problem of retraining and huge dataset requirements for each new hotword with good inference time on light-weight devices. To achieve the same we implemented Siamese neural network architecture with an image processing base network made with EfficientNet, which processes the Log Mel spectrograms of the respective input audio samples. Moreover, this network could also be repurposed for phrase detection's where a program needs to check for the occurrence of a specific sentence removing the requirement of heavy speech-to-text engines in edge devices. Such an engine can allow the end-users to set custom hotwords in their systems with minimal effort.

7. ACKNOWLEDGMENTS

References

[1] D. Amodei, R. Anubhai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, J. Chen, M. Chrzanowski, A. Coates, G. Diamos, E. Elsen, J. Engel, L. Fan, C. Fougner, T. Han, A. Hannun, B. Jun, P. LeGresley, L. Lin, S. Narang, A. Ng, S. Ozair, R. Prenger, J. Raiman, S. Satheesh, D. Seetapun, S. Sengupta, Y. Wang, Z. Wang, C. Wang, B. Xiao, D. Yogatama, J. Zhan, and Z. Zhu. Deep speech 2: End-to-end speech recognition in english and mandarin, 2015.

[2] K. Atsavasirilert, T. Theeramunkong, S. Usanavasin, A. Rugchatjaroen, S. Boonkla, J. Karnjana, S. Keerativittayanun, and M. Okumura. A light-weight deep convolutional neural network for speech emotion recognition using mel-spectrograms. pages 1–4, 10 2019.

[3] S. Becker, M. Ackermann, S. Lapuschkin, K.-R. Müller, and W. Samek. Interpreting and explaining deep neural networks for classification of audio signals, 2019.

[4] G. Chen and X. Yao. Snowboy hotword detection.

[5] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[6] S. Hershey, S. Chaudhuri, D. Ellis, J. Gemmeke, A. Jansen, R. Moore, M. Plakal, D. Platt, R. Saurous, B. Seybold, M. Slaney, R. Weiss, and K. Wilson. Cnn architectures for large-scale audio classification. pages 131–135, 03 2017.

[7] Y. Huang, T. Shabestary, and A. Gruenstein. Hotword cleaner: Dual-microphone adaptive noise cancellation with deferred filter coefficients for robust keyword spotting. pages 6346–6350, 05 2019.

[8] Y. Huang, T. Shabestary, A. Gruenstein, and L. Wan. Multi-microphone adaptive noise cancellation for robust hotword detection. pages 1233–1237, 09 2019.

[9] I. Kalith. Design and development of automatic speech recognition system for tamil language using cmu sphinx 4. 11 2012.

[10] J. Kaufman, J. Bathman, E. Myers, and W. Hingston. Google 10000 english words.

[11] A. Kenarsari. Rhino speech-to-intent engine: Context-aware nlu - picovoice, 2018.

[12] K. Kubota. Text-to-speech device and text-to-speech method, 06 2014.

[13] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar. Focal loss for dense object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP:1–1, 07 2018.

[14] A. Michaely, X. Zhang, G. Simko, C. Parada, and P. Aleksic. Keyword spotting for google assistant using contextual speech recognition. pages 272–278, 12 2017.

[15] C. Ooi, W.-H. Tan, S. Cheong, Y. L. Lee, V. Baskaran, and Y.-L. Low. Fpga-based embedded architecture for iot home automation application. *Indonesian Journal of Electrical Engineering and Computer Science*, 14:646–652, 05 2019.

[16] A. K. Picovoice, A. Kenarsari, D. Bartle, and R. P. Rostam. Picovoice/porcupine: On-device wake word detection powered by deep learning., Mar 2018.

[17] A. Reis, D. Paulino, H. Paredes, I. Barroso, M. Monteiro, V. Rodrigues, and J. Barroso. Using intelligent personal assistants to assist the elderlies an evaluation of amazon alexa, google assistant, microsoft cortana, and apple siri. pages 1–5, 06 2018.

[18] A. Saeed, D. Grangier, and N. Zeghidour. Contrastive learning of general-purpose audio representations. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3875–3879, 2021.

[19] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2015.

[20] M. Tan and Q. V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks, 2020.

[21] R. Tang, J. Lee, A. Razi, J. Cambre, I. Bicking, J. Kaye, and J. Lin. Howl: A deployed, open-source wake word detection system. In *Proceedings of Second Workshop for NLP Open Source Software (NLP-OSS)*, pages 61–65. Association for Computational Linguistics, Nov. 2020.

[22] R. Tang and J. Lin. Deep residual learning for small-footprint keyword spotting. pages 5484–5488, 04 2018.

[23] S. Team. Silero models: pre-trained enterprise-grade stt / tts models and benchmarks. https://github.com/snakers4/silero-models, 2021.

[24] M. Todisco, X. Wang, V. Vestman, M. Sahidullah, H. Delgado, A. Nautsch, J. Yamagishi, N. Evans, T. Kinnunen, and K. A. Lee. Asvspoof 2019: Future horizons in spoofed and fake audio detection. pages 1008–1012, 09 2019.

[25] F. Tom, M. Jain, and P. Dey. End-to-end audio replay attack detection using deep convolutional networks with attention. pages 681–685, 09 2018.

[26] A. Uitdenbogerd. International conference on music information retrieval 2003 [review article]. *Computer Music Journal*, 28:83–85, 06 2004.

[27] C. Vargas, Q. Zhang, and E. Izquierdo. One shot logo recognition based on siamese neural networks. pages 321–325, 06 2020.

[28] K.-P. Yang, A. Jee, D. Leblanc, J. Weaver, and Z. Armand. Experimenting with hotword detection: The pao-pal. *European Journal of Electrical Engineering and Computer Science*, 4(5), Sep. 2020.

[29] X. Yu, Z. Yu, and S. Ramalingam. Learning strict identity mappings in deep residual networks. pages 4432–4440, 06 2018.