Speaker Recognition using Convolutional Neural Network with Minimal Training Data for Smart Home Solutions

Mingshan Wang
Intel Corporation
San Diego, USA
mingshan.wang@intel.com

Tejaswini Sirlapu
Intel Corporation
San Diego, USA
tejaswini.sirlapu@intel.com

Alicja Kwasniewska
Gdansk University of Technology,
Faculty of Electronics,
Telecommunications and Informatics
Gdansk, Poland
alicja.kwasniewska@pg.edu.pl

Maciej Szankin
Intel Corporation
San Diego, USA
maciej.szankin@intel.com

Marko Bartscherer
Intel Corporation
San Diego, USA
marko.bartscherer@intel.com

Rey Nicolas
Intel Corporation
San Diego, USA
rey.nicolas@intel.com

***Abstract*—With the technology advancements in smart home sector, voice control and automation are key components that can make a real difference in people's lives. The voice recognition technology market continues to involve rapidly as almost all smart home devices are providing speaker recognition capability today. However, most of them provide cloud-based solutions or use very deep Neural Networks for speaker recognition task, which are not suitable models to run on smart home devices. In this paper, we compare relatively small Convolutional Neural Networks (CNN) and evaluate effectiveness of speaker recognition using these models on edge devices. In addition, we also apply transfer learning technique to deal with a problem of limited training data. By developing solution suitable for running inference locally on edge devices, we eliminate the well-known cloud computing issues, such as data privacy and network latency, etc. The preliminary results proved that the chosen model adapts the benefit of computer vision task by using CNN and spectrograms to perform speaker classification with precision and recall ~84% in time less than 60 ms on mobile device with Atom Cherry Trail processor.**

I. INTRODUCTION

Voice User Interface based smart home devices like the Amazon Echo [1] have become a household staple for interacting with the internet, controlling your house or shopping online. User authentication and security are important aspects of this usage. The most convenient way to provide the user authentication for voice User Interface (UI) based device is speech based speaker identification [2]. Moreover, to create a unique experience for each user in a single household, our smart home devices should possess the abilities to distinguish each individual through voices. The smart home of tomorrow uses artificial intelligence to understand how people engage with their homes and deliver seamless experiences, like identifying a known person of the household and adjusting room temperature automatically based on their preferences. The overview of a smart home setup for speaker recognition is shown in Fig.1. In order to maximize privacy and security, it is desirable to store the collected data and models that are used to uniquely identify the user on his or her personal device rather than sending it to a cloud service provider. At the same time, the cloud service providers like Amazon are facing bandwidth limitation and that migrates the computation back from the cloud to the edge devices. An additional advantage of these changes results in shortening response times, as all algorithms are running close to the user [1]. To support these requirements, we need to develop models and solutions that deliver the required accuracy while fitting into the memory and compute constrains of relatively low cost smart home devices. In case of portable battery powered devices, having low power consumption and minimal cost of data transfer are major concerns.

Existing solutions, e.g. Siri, Google Now, Cortana, S-Voice and Echo [1] have been in the speech recognition technology-market over the decade. They use speaker recognition technology usually deployed on the cloud [2]. However, these solutions have limitations when it comes to their application in a real-time enterprise environment that requires user privacy, security, and reliable connectivity. The major reason for improvement of both audio and image processing technologies was the recent progress of Deep Neural Networks (DNN) [3]. Over past few years, some attempts have been made to improve speaker recognition task with the means of deep learning [3]. Bhattacharya G. et al. [4] explored the DNN and Recurrent Neural Network (RNN) based approaches to recognize speakers. The DNN solution turns out to achieve better performance than RNN. Similar solution was described in [5]. In this study, authors compared

Fig.1. Speaker recognition scenario in a smart home setup; system recognizes a person by processing audio and reacts to commands with a personalized response, e.g. changes Air Conditioner temperature according to specific user's preferences

978-1-5386-5024-0/18/$31.00 ©2018 IEEE

Hidden Markov Models (HMM) with DNN. According to the results, HMM based method greatly improved the performance, which was also confirmed by [6]. DNN resulted in similar accuracy but required much more training samples. However, both presented works are text dependent, what significantly affects user experience as the models can be easily fooled and they require trigger words in order to work. Another approach that is also based on deep learning uses speaker discriminative CNNs to extract speech features. The features are then combined to form an utterance-level speaker vector through an attention mechanism. The proposed attention model takes the speaker discriminative information and the phonetic information to learn the weights for speaker verification task [7]. Due to fast deep learning progress, different neural network architectures have been proposed. It turns out that the deeper the model the better is the accuracy. However, models that currently achieve state-of-art results in classification task, e.g. Res-Net [8], are usually too big for memory requirement of smart home devices and require huge training data to give accurate results.

In this preliminary study, we investigate deep learning models for text independent speaker recognition suitable on smart home devices aiming for quick response time, less training data, low-power and high-accuracy speech recognition results. Our contribution lays in adapting Convolutional Neural Network (CNN), previously mainly used for image classification, to tackle speaker identification problem. In the study, we compare three CNN models, including a proposed three-layered network architecture. Additionally, we apply transfer learning from LibriSpeech [12] dataset to our collected dataset. Tests performed on Android device (Cherry Trail based Project Tango device) prove that the chosen architecture is suitable for smart home/wearable devices in terms of inference time while requiring only a small amount of training data. The accuracy of the proposed approach is verified with various people to address real-life smart home use-cases.

The paper is organized as follows: Section II presents three neural network architectures based on convolutional operations that are suitable for low power devices. In Section III we describe preliminary results that include precision, recall, F1 scores and inference time. In Section IV we present the discussion of the significances from our results. Section V concludes the paper and provides directions for future work.

II. METHODOLOGY

In this section, we briefly provide the motivation for using CNN-based network architecture, followed by a detailed description of all tested models. Then we state our data collection procedure and we will discuss the details of our training procedure, including the transfer learning we adapt. At the end, we describe our testing experiments.

A. Architecture

Our goal was to develop a model that is able to run on smart home devices, which means it needs to have low computational power usage and small memory footprint. In our experiments, we tested three CNN architectures:

1. *NN1 – 2-conv layer model, motivated by architecture described in [9], which was originally designed for keyword spotting on mobile devices. The detailed architecture of this model is presented in Fig. 2.*
2. *NN2 - Modification of the NN1 – one additional convolute*
3. *onal layer, followed by ReLu and max pooling operation inserted between layer 4 and 5 from Fig. 2.*
4. *NN3 - SqueezeNet v1.1 [10] designed for low power devices, architecture presented in details in Fig. 3.*

ID	Name	Type	Channel input	Dimension input	Channel output	Dimension output	Operations	Memory
1	data	InputReshaped	1	55x40	1	55x40		activation 2.2k
2	conv1	Convolution	1	55x40	64	55x40	macc 22.53M	activation 140.8k
								param 10.3k
3	relu_conv1	ReLU	64	55x40	64	55x40	comp 140.8k	activation 140.8k
4	pool1	Pooling	64	55x40	64	28x20	comp 143.36k	activation 35.84k
5	conv2	Convolution	64	28x20	64	28x20	macc 91.75M	activation 35.84k param 163.9k
6	relu_conv2	ReLU	64	28x20	64	28x20	comp 35.84k	activation 35.84k
7	reshape	Reshape	64	28x20	35840	1x1		
8	matmul	InnerProduct	35840	1x1	6	1x1	macc 215.04k	activation 6 param 215.05k
9	prob	Softmax	6	1x1	6	1x1	add 6	activation 6
							div 6	
							exp 6	
	TOTAL						macc 114.49M comp 320k add 6 div 6 exp 6	activation 391.33k param 389.25k

Fig.2. Architecture, number of operations and parameters of the tested 2-conv layer network (NN1)

In our research, CNN is chosen, as it is naturally good with acoustic sound modeling since it considers the relationship in a local filter area in terms of time and frequency [13]. Moreover, in previous studies it has been proved to produce high accuracy for speaker recognition task [10].

B. Data Collection

We collected all the voice data using the Apple Earpods plugged into a computer where a running Python script processed, and saved the recordings for four different speakers. The Python script used to process the recording actively listens to the speaking from a speaker, stops listening and saves the recording when there is no sound. The script saves each recording to a 16 kHz wav-format audio file, which is the required format for our model training procedure. In total, we have conducted two rounds of data collection. In the first round, each individual repeated speaking 'Hi there, it's me' for 100 times.

In the second round each individual spoke the 10 different commands below, starting with a trigger word 'Hi there':

- Hi there, play some music.
- Hi there, what is the time now?
- Hi there, how is the weather today?
- Hi there, what is on my calendar?
- Hi there, what is on my reminder?
- Hi there, ring my phone.
- Hi there, turn on the lights.
- Hi there, what is on my shopping list?
- Hi there, connect to my phone.
- Hi there, turn on the TV.

For each command, the speaker repeated speaking for 20 times. In total, we recorded 200 samples per person in the second round. All the voice data is saved as a wav file format with frequency 16 kHz. Each of the recording in the first round is 1 second long, and the individual recording in the second round is 3 seconds long. Hence, our training set consists of 1200 samples, 300 for each of speakers.

Although we have only collected four speaker's audio data, our final prediction categories have two additional categories: silence and unknown, giving total six possible output labels. These two additional categories are always added during the training. This setup is derived from the Tensorflow Simple Audio Recognition tutorial [15]. For unknown category, the speaker folders that are not selected for identification will be automatically treated as unknown category. If all speakers' folders are selected for identification, then there will be no samples used as unknown category. As for silence category, it is added to simulate silence situation in real life where no speaker is speaking but some noise still can be heard. The samples used as silence category come from a 'background_noise' folder, which we found from the Speech Commands Dataset [11]. During the training, we selected 10% of the samples from the 'background_noise' folder to become silence category.

In order to take advantage of CNN in image processing, we converted wav audio files into spectrograms and then used the spectrogram as an input to the CNN pipeline. A spectrogram is essentially a visual representation of an acoustic signal. Examples of input spectrogram for each speaker using the same voice command "Hi there, it's me" are presented in Fig. 4. For spectrograms in Fig. 4, the vertical axis represents time, increasing from top to bottom, where the horizontal axis represents frequency, reading from left to right. The velocity of pronouncing the given phrase is another voice characteristic that distinguish various speakers. Therefore, spectrograms were not squeezed or extended on the time axis in order to preserve this feature.

ID	Name	Type	Channel input	Dimension input	Channel output	Dimension output	Operations	memory
1	data	Input	1	227x227	1	227x227		activation 515.29k
2	conv1	Convolution	1	227x227	64	113x113	macc 73.55M	activ. 8.17M param 640
3	relu_conv1	ReLU	64	113x113	64	113x113	comp 8.17M	activation 8.17M
4	pool1	Pooling	64	113x113	64	56x56	comp 18.06M	activation 2.01M
5	fire2	submodule(6)	64	56x56	128	56x56	macc 353.24M comp 4.52M	activ. 13.05M param 11.41k
12	fire3	submodule(6)	128	56x56	128	56x56	macc 385.35M comp 4.52M	activ. 13.05M param 12.43k
19	pool3	Pooling	128	56x56	128	28x28	comp 9.03M	activation 1M
20	fire4	submodule(6)	128	28x28	256	28x28	macc 353.24M comp 2.26M	activ. 6.52M param 45.34k
27	fire5	submodule(6)	256	28x28	256	28x28	macc 385.35M comp 2.26M	activ. 6.52M param 49.44k
34	pool5	Pooling	256	28x28	256	14x14	comp 4.52M	activation 501.76k
35	fire6	submodule(6)	256	14x14	384	14x14	macc 204.72M comp 846.72k	activ. 2.45M param 104.88k
42	fire7	submodule(6)	384	14x14	384	14x14	macc 216.76M comp 846.72k	activ. 2.45M param 111.02k
49	fire8	submodule(6)	384	14x14	512	14x14	macc 369.3M comp 1.13M	activ. 3.26M param 188.99k
56	fire9	submodule(6)	512	14x14	512	14x14	macc 385.35M comp 1.13M	activ. 3.26M param 197.18k
63	drop9	Dropout	512	14x14	512	14x14	comp 1M	activation 1M
64	conv10	Convolution	512	14x14	6	14x14	macc 6.02M	activation 11.76k param 3.08k
65	relu_conv10	ReLU	6	14x14	6	14x14	comp 11.76k	activation 11.76k
66	pool10	Pooling	6	14x14	6	1x1	add 11.76k	activation 60
67	prob	Softmax	6	1x1	6	1x1	add 60 div 60 exp 60	activation 60
	TOTAL						macc 2.73G. comp 58.3M add 11.82k div 60 exp 60	activation 71.95M param 724.42k

Fig. 3. Architecture, number of operations and parameters of the tested SqueezeNet network (NN3)

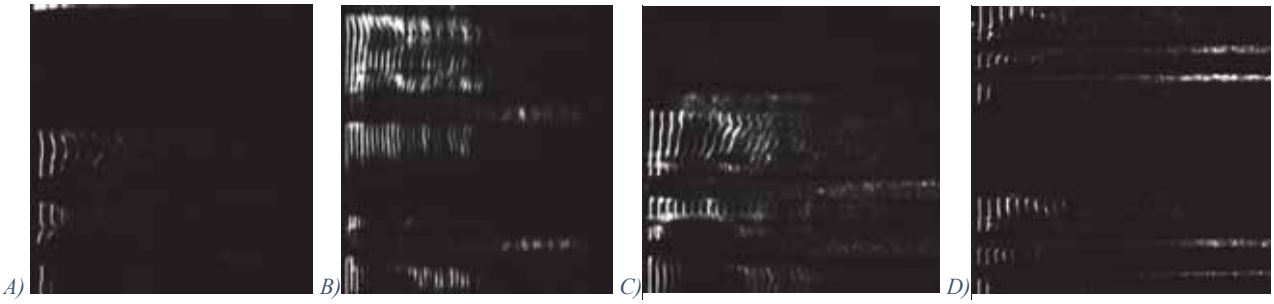

Figure 4. Spectrograms generated for speaker A, B, C, D saying the same sentence 'Hi there, it's me'

C. Training Procedure

The code we used for training is adapted from the Tensorflow Simple Audio Recognition tutorial [15]. Since the spectrograms are used for training the neural network, we need to properly generate the spectrograms from the audio files so that the spectrograms contain the maximized useful information in them. To be able to do so, we need to experiment different combination of window size value and window stride value, which both defined in millisecond (ms), which are used to generate the spectrogram. The window size value controls how long each spectrogram time slice is, and the window stride value controls how much overlap between each spectrogram time slices. If the window size is too big for a given sample, the spectrogram for that sample will lose the temporal resolution in the original audio, on the other contrary, if the window size is too short, it will not able to capture enough information in the original audio. Another way to capture more temporal resolution is to have shorter window stride value, but it may lead to higher computation cost. That's to say, we want to find a right combination of window size and window stride value so that the spectrograms generated from the training audio samples will contain enough temporal information for the CNN model to capture and learn. Therefore, spectrogram generation is the first important step leading to a good model. After trying out different values, the best parameter set for our collected samples is window size of 1024 ms and window stride of 64 ms. For the LibriSpeech samples, the best parameter combination is window size of 1024 ms and window stride of 256 ms. For both the 2-layered and 3-layered CNN model, we all use the above parameters to generate the corresponding spectrograms for training.

The other parameters we set during training 2-layered and 3-layered models for both collected dataset and LibriSpeech Dataset are as follow: train/validation/test set proportion of 70:15:15, dropout {0.5}, learning rate {0.001, 0.0001}, training steps {15000, 3000}. The batch size for our collected dataset is 5 and for LibriSpeech Dataset is 100, since our collected dataset have much less samples compared to LibriSpeech Dataset, so we lowered the batch size value accordingly during the training. The 2-layered models and 3-layered models trained on our collected dataset with the above parameters both gave 100% training accuracy and 99.2% test accuracy. The 2 layered-model trained on LibriSpeech dataset gave 95% training accuracy and 93% test accuracy at the end of the training, and the 3-layered model trained on LibriSpeech Dataset gave 90% training accuracy and 88% test accuracy.

SqueezeNet model [14] is a relatively small (~4.7MB) neural network, which achieves accuracy equivalent to bigger models, e.g. AlexNet [11]. In our research we started by training SqueezeNet model on spectrograms from LibriSpeech [12] dataset. Data was divided into training, validation and test sets in proportion 70:15:15. At first, we tested hyperparameters from the SqueezeNet repository (see Table 1). Then, we introduced various modifications to explore how this model performs if trained with other parameters. Combinations of hyperparameters with following values were examined: learning rate {0.04, 0.01, 0.001}, weight decay {0.002, 0.005}, learning rate decrease policy {polynomial with power 1.0}, momentum {0.8, 0.9}, iterations {1000-12000, tested with a step 1000}, optimizer {Stochastic Gradient Descent}.

TABLE I. HYPERPARAMETERS FROM SQUEEZENET REPOSITORY

SqueezeNet v1.1. hyperparamets	
Parameter	***Value***
Learning rate	0.04
Learning policy	Polynomial
Power	1.0
Momentum	0.9
Weight decay	0.0002
Optimizer	Stochastic Gradient Descent

It turned out that best results were achieved for learning rate 0.01, learning policy polynomial with power 1.0, momentum 0.9 and weight decay 0.0002. After 12000 iterations loss was reduced to 0.099, producing accuracy 96.88% on the training set and 97.62% on the validation set.

Next, we tested combinations of the same hyperparameters during training SqueezeNet model v.1.1 on our dataset. Our dataset was also divided into training, validation and test sets in proportion 70:15:15. In this case, the best results were achieved for learning rate 0.04, learning policy polynomial with power 1.0, momentum 0.9 and weight decay 0.0002, yet the final loss, even after 12000 was high (1.38) and accuracy both on train and validation very low (34.4% and 25.0% on train and validation sets respectively). This was probably caused by a small number on samples in our dataset, that did not allow the model to learn proper correlations.

D. Transfer Learning

Since we have limited amount of training data, in order to further improve model accuracy, we tried transfer learning technique. In real-world applications, we usually want to solve a task in one domain of interest, but we may lack of the data for that specific domain, however, we could have sufficient training data in another domain of interest. In this case, the concept of knowledge transfer between different domains emerges, that's to say, we can transfer the knowledge learned from dataset A to help the training of dataset B, where dataset A and dataset B doesn't need to come from the same domain, and dataset A may have much larger size than dataset B. This idea of knowledge transfer enabled transfer learning in deep learning model training, and transfer learning is proved helpful for classification, regression and clustering problems, given the correct training procedure [12]. For our case, we mainly want to solve the problem of running speaker recognition model on edge devices in a household, but we do not have a huge dataset of audio recordings from edge devices. Therefore, we found the LibriSpeech Dataset [12], which is a corpus of audio book recordings. In this dataset, each speaker was reading an audiobook, and the recordings for a single audio book are divided into 15 seconds samples.

To apply transfer learning, firstly we trained all the models we want to compare on a bigger dataset, LibriSpeech dataset [12] in this case. Then for two-layered model and the three-layered model, we reused all the weights before the fully connected (FC) layer, and then retrained the weights for the FC layer using our collected dataset.

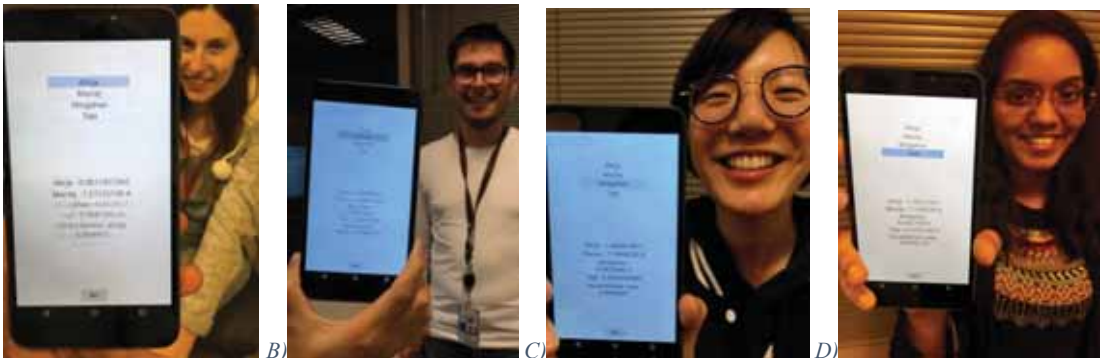

Fig. 5. Text independent speaker recognition results (highlighted name of the prediction) during live experiments with Android application

All the training parameters stated in the previous section stays the same in transfer learning training procedure. Transfer learning technique was also applied in the same manner (restore all weights before FC layer and retrain FC layer using our data) to SqueezeNet model for our dataset, because after training the network from scratch on our samples, it turned out that the amount of training data is too small to achieve a satisfactory accuracy (34.4% and 25.0% on train and validation sets respectively). The size of LibriSpeech database, though, was much bigger and therefore, we explored if the neural network trained on LibriSpeech, used to finetune the model to our dataset, will produce higher accuracy. In this case, learning rate was reduced to 0.0001, as bigger values of this hyperparameter led the model to stay in a local optima instead of a global. As a result of applying transfer learning technique to SqueezeNet model retrained on our dataset, the accuracy was increased by 62.5% (34.4% to 96.9%) for training set and by 75% (25.0% to 100%) for validation set in only 1000 iterations instead of 12000 as for learning from scratch.

III. RESULTS

The metrics we used to measure the performance of the models are precision, recall and F1 score. To validate models, we collected a separate test set that consists of 50 samples per speaker. Each speaker repeated 10 commands, which are different from the commands used for training, 5 times each. The 10 commands we used for testing are:

- Hi there, mute.
- Hi there, unmute.
- Hi there, show my calendar.
- Hi there, show me pictures of cats.
- Hi there, show me movie show times at AMC.
- Hi there, next.
- Hi there, order food.
- Hi there, wake me up at 7 am.
- Hi there, set a timer for 10 mins.
- Hi there, what's on my shopping list.

In addition to tests performed on recorded test dataset, we also developed an Android application for performing live tests. The mobile environment allowed us to stimulate smart home scenarios. To prove that the model we chose are suitable to run on edge devices we measured the inference time of processing 10 samples per speaker on Project Tango device with Cherry Trail processor. Table II shows results achieved on LibriSpeech database, while training each model from scratch. Table III presents the same metrics, but for training from scratch using our dataset. In Table IV we present performance metrics on the test set after applying transfer learning technique in order to finetune each network architecture from LibriSpeech dataset to our data. Inference time for running each model on Project Tango device using batch size equal 1 is shown in Table V. For inference time measurement we used 10 samples per each speaker during live experiments. Table VI and Table VII shows the confusion matrix of running the two-layered model (NN1) and the three-layered model (NN2) respectively on our recorded test dataset.

The images in Fig. 5 are results of our live testing using the Android application we prepared based on Tensorflow speech recognition solution [15]. Our Android application collects input from the microphone on the phone, then runs the model classification on the device, at the end the predicted speaker name is highlighted. The table below speaker name list displays the confidence score for each person. In all the result tables, NN1 refers to the two-layered CNN model, NN2 refers to our modified three-layered CNN model, and NN3 refers to the SqueezeNet model.

TABLE II. VALIDATION METRICS - 15% SAMPLES FROM LIBRISPEECH

Model Name	**Training from Scratch - LibriSpeech Dataset**		
	Average Precision	***Average Recall***	***Average F1 Score***
NN1	0.917	0.922	0.917
NN2	0.890	0.890	0.885
NN3	0.16	0.22	0.15

TABLE III. VALIDATION METRICS - OUR COLLECTED TEST SET – TRAINING FROM SCRATCH

Model Name	**Training from Scratch - our dataset**		
	Individual Precision	***Individual Recall***	***F1 Score***
NN1	Alicja: 0.80, Maciej: 0.88, Teja: 0.76, Mingshan: 0.73 **Avg: 0.79**	Alicja: 0.72, Maciej: 1.00, Teja: 0.58, Mingshan: 0.88 **Avg: 0.80**	Alicja: 0.76, Maciej: 0.93, Teja: 0.66, Mingshan: 0.8 **Avg: 0.79**

Model Name	Training from Scratch - our dataset		
	Individual Precision	*Individual Recall*	*F1 Score*
NN2	Alicja: 0.87, Maciej: 0.86, Teja: 0.82, Mingshan: 0.79 **Avg: 0.84**	Alicja: 0.8, Maciej: 1.0, Teja: 0.54, Mingshan: 1.0 **Avg: 0.84**	Alicja: 0.83, Maciej: 0.93, Teja: 0.65, Mingshan:0.89 **Avg: 0.82**
NN3	Too few samples to train model that will achieve satisfactory performance; maximal accuracy on the train set 34.4%; 25% on the validation set; **0% on a test set**		

TABLE IV. VALIDATION METRICS - OUR COLLECTED TEST SET – TRANSFER LEARNING

Model name	Transfer Learning – from LibriSpeech to our dataset		
	Individual Precision	*Individual Recall*	*F1 Score*
NN1	Alicja: 0.42 Maciej: 0.10 Mingshan: 0.0 Teja: 0.25 **Avg: 0.19**	Alicja: 0.2 Maciej: 0.06 Mingshan: 0.0 Teja: 0.58 **Avg: 0.21**	Alicja: 0.27 Maciej: 0.08 Mingshan: 0.0 Teja: 0.35 **Avg: 0.18**
NN2	Alicja: 1.0 Maciej: 0.045 Mingshan: 0.19 Teja: 0.0 **Avg: 0.31**	Alicja: 0.24 Maciej: 0.02 Mingshan: 0.16 Teja: 0.0 **Avg: 0.105**	Alicja: 0.39 Maciej: 0.028 Mingshan: 0.174 Teja: 0.0 **Avg: 0.148**
NN3	Alicja: 0.89 Maciej: 0.98 Mingshan: 0.87 Teja: 0.94 **Avg: 0.92**	Alicja: 0.87 Maciej: 0.98 Mingsan: 0.82 Teja: 1.0 **Avg: 0.92**	Alicja: 0.88 Maciej: 0.98 Mingshan: 0.84 Teja: 0.97 **Avg:0.92**

TABLE V. AVERAGE INFERENCE TIME DURING LIVE EXPERIMENTS; 10 SAMPLES PER SPEAKER; BATCH SIZE = 1 ON PROJECT TANGO DEVICE [MS]

Inference time [ms]		
NN1	*NN2*	*SqueezeNet (NN3)*
57.7 ± 5.6	59.6 ± 7.8	360.82 ± 76.82

We also created a confusion matrix from our testing for analysis. The column name in the confusion matrix represents the true label, and the row name represents the predicted label. Therefore, the perfect model will have all zeros in the confusion matrix except the diagonal.

TABLE VI. CONFUSION MATRIX FOR 2 LAYERED MODEL (NN1) TRAINING FROM SCRATCH ON OUR TEST DATASET

	Confusion Matrix			
	Alicja	*Maciej*	*Mingshan*	*Teja*
Alicja	36	4	1	9
Maciej	0	50	0	0
Mingshan	6	0	44	0
Teja	3	3	15	29

TABLE VII. CONFUSION MATRIX FOR 3 LAYERED MODEL(NN2) TRAINING FROM SCRATCH ON OUR TEST DATASET

	Confusion Matrix			
	Alicja	*Maciej*	*Mingshan*	*Teja*
Alicja	40	4	0	6
Maciej	0	50	0	0
Mingshan	0	0	50	0
Teja	6	4	13	27

IV. DISCUSSION

In our study, we evaluated and compared the performance of deep learning models suitable for low power edge devices in a task of speaker recognition. Precision is calculated as:

$$Precision = \frac{TP}{TP+FP} \quad (1)$$

where TP indicates True Positives and FP - False Positives. It represents for each person, how much percentage of the predicted samples actually belong to the predicted person. This is an important metric to look at if we want to enable authentication or parental control on smart home devices, since we do not want unknown person to make transaction using our identities, or our children access prohibited content accidentally. Recall is calculated as:

$$Recall = \frac{TP}{TP+FN} \quad (2)$$

where FN indicates False Negatives. It measures for each person, how likely our model is able to distinguish him/her from the other speakers in the database. Relatively low recall value may be acceptable for smart home scenarios because it may just add some extra time for users to access some services, but it will not result in any critical consequences. F1 score is calculated as a balanced measure of precision and recall:

$$F1\ score = \frac{2Precision*Recall}{Precision+Recall} \quad (3)$$

From Table II, we can see that when trained on LibriSpeech Dataset, 2-layered CNN model (NN1) achieve a little bit higher average precision, recall and F1 score than our modified 3-layered CNN model (NN2). We believe this difference can be caused by overfitting on the training dataset for the NN2. Generally, more layers in a CNN architecture will lead to better classification accuracy, but since our model is relatively simple, when trained with the same set of hyperparameters, the 3-layered model overfits while the 2-layered model is not, so the performance of the 2-layered model is better than the 3-layered model on LibriSpeech Dataset. In case on SqueezeNet model, the results are much worse. Although we tested different hyperparametrs and selected the most prominent ones, the model was not able to generalize well to new data.

Table III shows that the 3-layered model achieves higher avg. precision, recall and F1 score than the 2-layered model, although individual precision and recall have varied a little bit. In addition, looking at Table VI and VII together, we can tell that for 2 speakers (Mingshan and Alicja), the 3-layered model improves a lot and end up predicting most or even all test samples correctly. Our dataset had too few samples to train SqueezeNet model that will achieve satisfactory performance. Although, various combinations of hyperparameters were tested, the maximal accuracy on the train set was 34.4%; and 25% on the validation set.

In case of SqueezeNet model, as can be seen in Table IV, transfer learning led to significant performance improvement of a model retrained on samples collected by us, although the size of our dataset was very small. Also, the training time was greatly reduced (from 12000 iterations to 1000), which indicates that this solution is suitable for smart devices, where user is not required to spend much time of setting up the device and collecting samples. On the contrary, both the 2-layered and 3-layered CNN models are not improving at all with

transfer learning. This could be caused by the shallow CNN architecture that overfits on the LibriSpeech Dataset and is not able to generalize on our collected dataset, since from Table II we can tell that our model gave decent accuracy when training from scratch with LibriSpeech dataset.

Another interesting thing to observe is that for one of the speakers (Maciej), the confusion matrix entry for both 2-layered model and the 3-layered model is perfect, since for all 50 samples, both models produced 100% correct predictions. The reason is probably that he is the only male speaker in our dataset, which makes his voice more distinct from the other three female speakers. In the future, we want to experiment with same percentage of male and female voices.

In this study, we proposed a text-independent speaker recognition that is more difficult task than text-dependent classification in that it requires more information to be able to capture the important features from individual speaker's acoustic fingerprint in general instead of features when he/she speaks specific text. From the results we obtained, CNNs is possible to capture the important feature for different speakers. The presented approach can be implemented on smart home low power devices, as it fulfills the requirements of small memory footprint and short processing time while running inference only locally. However, the achieved results are only preliminary and should be further confirmed by collecting audio samples of more volunteers. Some future work can be explored, e.g. how to generate more unique speaker relevant feature in the spectrogram and recognize more people.

V. CONCLUSION

The conducted research was focused on evaluating various DNN models used for recognizing speakers using spectrograms generated from audio files on mobile devices. From preliminary results, we observe that simple and relatively shallow (2 and 3 convolutional layers) CNNs achieve decent accuracy (precision and recall ~84%) with minimal training data (300 samples per speaker) and they are feasible to run on edge devices, because of their small memory footprint (3.5 MB) and short inference time (~58 ms on Atom CherryTrail processor). SqueezeNet model was also satisfactory in terms of model size (4.7 MB), however, the inference time was much higher (~360 ms). Also, both on our samples and LibriSpeech dataset, if trained from scratch, was not able to generalize well to new samples, producing very low (<0-22>%) precision and recall. On the other hand, transfer learning technique used to finetune SqueezeNet model trained on LibriSpeech dataset to our samples, significantly improved the model accuracy, producing precision and recall equal 0.92. This proves, that even if a user of a smart home setup wants to spend minimal time for model retraining, it is possible, but providing previously trained model on a bigger database. The presented study enables possibilities for future smart home visions while being suitable for resource-constrained devices and is aligned with the general trend to move the computation from the cloud back to the edge devices. In future work, we would like to focus on evaluating performance on more samples and enabling the proposed solution on platforms that recognize more speakers, e.g. smart conference room, where model dynamically controls the presentation content based on the detected speaker.

ACKNOWLEDGMENT

This work has been partially supported by Intel Corporation, USA. We thank all our colleagues from Intel Corp. who provided insight and expertise that greatly assisted the research. It has been also partially supported by Statutory Funds of Electronics, Telecommunications and Informatics Faculty, Gdansk University of Technology.

REFERENCES

[1] Mehdi Assefi, Guangchi Liu, Mike P. Wittie, Clemente Izurieta Department of Computer Science, Montana State University, "An Experimental Evaluation of Apple Siri and Google Speech Recognition"

[2] Doddington, G.R., 1985. Speaker recognition—Identifying people by their voices. Proceedings of the IEEE, 73(11), pp.1651-1664.

[3] Yu, W., Liang, F., He, X., Hatcher, W.G., Lu, C., Lin, J. and Yang, X., 2018. A Survey on the Edge Computing for the Internet of Things. IEEE Access, 6, pp.6900-6919.

[4] Haack, W., Severance, M., Wallace, M. and Wohlwend, J., 2017. Security Analysis of the Amazon Echo, Accessed: March, 27th, 2018; available: https://courses.csail.mit.edu/6.857/2017/project/8.pdf

[5] Goodfellow, I., Bengio, Y., Courville, A. and Bengio, Y., Deep learning (Vol. 1). 3rd ed. Cambridge: MIT press, 2016.

[6] Erik McDermott2, Ignacio Lopez Moreno2 , Javier Gonzalez-Dominguez2, MD USA 2Google Inc., "Usa Deep Neural Networks For Small Footprint Text-Dependent Speaker Verification"

[7] Bhattacharya, G., Alam, J., Stafylakis, T. and Kenny, P., 2016. Deep neural network based text-dependent speaker recognition: Preliminary results. Odyssey 2016, pp.9-15

[8] Zeinali, H., Burget, L., Sameti, H., Glembek, O. and Plchot, O., 2016, June. Deep neural networks and hidden Markov models in i-vector-based text-dependent speaker verification. In Odyssey-The Speaker and Language Recognition Workshop (pp. 24-30)

[9] Hossein Zeinali, Hossein Sameti, and Lukas Burget, "HMM-based phrase-independent i-vector extractor for text-dependent speaker verification," Audio, Speech, and Language Processing, IEEE Transactions on, in.press.

[10] Zhang, S.X., Chen, Z., Zhao, Y., Li, J. and Gong, Y., 2016, December. End-to-end attention based text-dependent speaker verification. In Spoken Language Technology Workshop (SLT), 2016 IEEE (pp. 171-178). IEEE.

[11] He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[12] LibriSpeech ASR corpus: available: http://www.openslr.org/12/ Accessed March, 29th 2018.

[13] Sainath, T.N. and Parada, C., 2015. Convolutional neural networks for small-footprint keyword spotting. In Sixteenth Annual Conference of the International Speech Communication Association.

[14] landola, F.N., Han, S., Moskewicz, M.W., Ashraf, K., Dally, W.J. and Keutzer, K., 2016. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size. arXiv preprint arXiv:1602.07360.

[15] Tensorflow Simple Audio Recognition tutorial, Accesed: Feb. 12th, 2018 www.tensorflow.org/versions/master/tutorials/audio_recognition

[16] Speech Command Dataset, Google, Accessed: Feb. 5th, 2018, available: https://research.googleblog.com/2017/08/launching-speech-commands-dataset.html

[17] Krizhevsky, A., Sutskever, I. and Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).

[18] S. J. Pan and Q. Yang, "A Survey on Transfer Learning," in *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345-1359, Oct. 2010.