Improvement of a CNN simple audio recognition system

Alessio Stefanelli[†], Giulio Pilotto[‡]

***Abstract*—Speech interfaces are spreading over every different types of digital applications, because they make the interaction easier,requiring less background knowledge, than using a tangible interfaces. In this paper we study and improve a Keyword Spotting System (KWS) implemented using a CNN architectures already presented in [1] and implemented by Tensorflow team under the section "Speech recognition tutorial". We improved the original model with different techniques, our model achieve 92% of accuracy respect to the 87% achieved by the original model, without increasing drastically the number of the parameters. Moreover we expand our study measuring precision and accuracy, using the android application provided in the source code by Tensorflow team, comparing performances between the original model and our improved models.**

***Index Terms*—**

I. Introduction

Keyword Spotting (KWS) are largely use nowadays in order to simplify and make the human-computer interaction easier. Moreover it is used to speed up the learning process and make accessible to everyone all the digital artifacts independently from physical abilities, culture and educational level. Examples take into account all the free-hand applications like driver assistant, home automation and also critical event management like in space mission. KWS systems are requested to make the interaction with the device less frustrating than reading and understanding the whole instructions' book. This can be actuated only if the system detects in real time the word pronounced, and carries out the desired action. The real time constraint requires the system to run locally, in order to provide as fast as possible the wanted action. The second request is to be accessible to every native speaker or adopted, independently from the pronunciation,for example in Italy exist more than 20 different dialect that make possible to have 20 different pronunciation of the same word. It also has to be portable in order to be stored in devices like smart-phone or board, like raspberry Pi, that doesn't have high computational power; in fact one of the limits of KWS is that it's implemented with complex models made of many parameters that makes the system slower and resource consuming.

The earliest approaches to speech recognition were based on finding speech sounds and providing appropriate labels to them. The techniques were

- Acoustic-method approach: is based on the theory of acoustic phonetics and postulates; here it is assumed that the rules governing the variability are straightforward and can be readily learned by a machine. There are three techniques that have been applied to the language identification: Problem-phone recognition, Gaussian mixture modeling, and Support vector machine classification [2] [3].
- Pattern-matching approach: involves two essential steps namely pattern training and pattern comparison [4]. In the pattern-comparison stage of the approach, a direct comparison is made between the unknown speeches (the speech to be recognized) with each possible pattern learned in the training stage in order to determine the identity of the unknown word according to the goodness of match of the patterns.
- Template based approaches: unknown speech is compared against a set of pre-recorded words (templates) in order to find the best Match [5]. This has the advantage of using perfectly accurate word models,but template preparation and matching become prohibitively expensive or impractical as vocabulary size increases beyond a few hundred words.
- Expert knowledge approach: manually codes into a system the speech variation [6]. This has the advantage of explicit modeling the variations in speech; but unfortunately such expert knowledge is difficult to obtain and use successfully.

Lately, temporal and spectral variability have been taken into account in speech recognition:

- Statistical based approach: variations in speech are modeled statistically, using automatic statistical learning procedure, typically the Hidden Markov Models, or HMM [7].The main disadvantage of statistical models is that they must take a priori modeling assumptions which are answerable to be inaccurate, handicapping the system performance. Compared to template based approach, hidden Markov modeling is more general and has a firmer mathematical foundation.
- Artificial Intelligence approach: is a hybrid of the acoustic phonetic approach and pattern recognition approach. In this, it exploits the ideas and concepts of Acoustic phonetic and pattern recognition methods [8].
- Stochastic modeling: [9] entails the use of probabilistic models to deal with uncertain or incomplete information.The most popular stochastic approach today is hidden Markov modeling.

Convolutional Neural Networks (CNNs) have become popular for acoustic modeling in the past few years, showing

[†]UNIPD, email: {alessio.stefanelli}@dei.unipd.it
[‡]UNIPD, email: {pilottogiu}@dei.unipd.it

Special thanks to Tensorflow documentation and source codes.

improvements over Deep Neural Networks (DNNs) which implement HMM system for speech recognition. [1]. In this paper we want to exploit a ready-to-use CNN published in the Tensorflow website under the speech-recognition section, used to implement a KWS system. The standard model presented in Tensorflow speech-recognition example, achieves an accuracy of 87%; we show that it can be augmented, without increasing drastically the number of parameters. Another aim of our research is to find out how many layers are necessary in order to pursuit cost-effectiveness in terms of numbers of parameters and accuracy. Playing around ADAM learning and regularization we achieved better results, finding out an implementation with less parameters and multiplication that achieve an accuracy near the original model. Moreover adding a layer does not improve accuracy meaning that 1 or 2 layers are enough and more layers will mess the features-space. We organize the rest of the paper as follow. Section 2 briefly describe feature work, Section 3 contains a processing pipeline of the blocks that we have developed. Section 4 describes the dataset we use and the processing operation for feature extraction, in Section 5 we present all the different models that we implemented, in Section 6 we present the result and in Section 7 we make our conclusion over this work, in Section 8 we state what we learn developing this project.

II. Related Work

A. HMM-GMM approach

In order to recognize speech, we need a classifier able to identify which (if any) phoneme was uttered in every audio frame. A simple Gaussian Mixture Model could be use for that, i.e. for each phoneme class; the classifier fits a GMM using all the frames where that phoneme is found - in other words, to classify a new utterance, we would check frame by frame which phoneme is the most probable. However,this kind of model does not exploit the temporal dependencies in the acoustic signal - the classification depends only upon the current frame, ignoring the context (i.e. previous and next frames). Moreover, this model assumes that there is no acoustic difference in the beginning, middle and ending of a phoneme. The GMM-HMM model is a response to these problems. The HMM is a temporal model, which assumes that the source has some state which we do not know about (e.g. position of the larynx, shape of the oral cavity, tongue placement).

Most common HMM architectures for speech recognition have phoneme models consisting of three states: in this way it is assumed that a phoneme, when uttered, has three distinct phases - a beginning, a middle, and an ending part and, in each phase, it sounds a bit different. The confidence of the keyword is computed as the likelihood ratio of each state.

$$C_{kw} = L_{left} + L(KW) + L_{right} - L_{bkg}$$

L_{bkg} is the likelihood of the utterance without considering the keyword. In [10] two HMM-GMM model are presented and trained with two different database using two different feature extraction techniques. In ICS10h they used 13 Mel-frequency cepstral coefficient with double delta and in ICS277h 13 perceptual linear prediction (PLP) coefficient with double deltas. Both systems use crossword tri-phone models which are context sensitive. In the paper the authors use Figure of Merit (FOM) as a measure of evaluation, which is an average of correct detection per numbers of false alarm, per hour. Results show that ICS277h perform better with 63.66% FOM, but the number of parameters and multiplication where too high considering the tri-phone context structure, having, in this way, a system hardly portable and real-time. A different HMM model is the Keyword-Filler model, which showed an important advantage sice it does not require keyword-specific data at training-time; it simply learns a generative model for all tri-phone HMM states through likelihood maximization on general speech data [11].

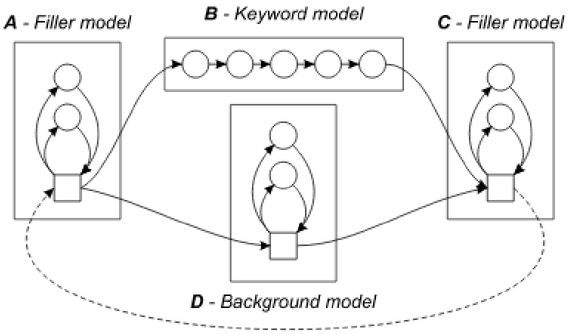


Fig. 1: HMM Keyword/filler model

B. DNN approach

The GMM-HMM model presented before, was less competitive than the TRAP-NN40h model, a phoneme recognizer based on temporal patterns (TRAPs) and neural networks (NN), this system makes use of a feature extraction technique based on long temporal trajectories: the temporal context of critical band spectral densities is splitted into left and right context. This allows a more precise modeling of the whole trajectory while limiting the size of the model (number of wights). Then, both parts are processed by DCT (Discrete Cosine Transform) and the feature vector is fed to NN. In a third step the NN works as a merger and produces final set of phoneme posterior. The advantage of this system is also the efficiency of the recognition network, on the other hand when a new keyword is entered, this system must go through all the data which can make the search time impractical. In [12] a discriminative KWS approach based is presented: a DNN is trained in order to directly predict keywords and then is followed by a posterior handling method producing a final confidence score.In contrast with HMM approach, this does not require a sequence algorithm (decoding) leading to a significantly simple implementation. The proposed DNN KWS system outperform a HMM Keyword-Filler model even when it is trained with less data and fewer parameters. In addition DNN is attractive to run on the device, as the size of the model can be easily adjusted by changing the number of parameter on the network.

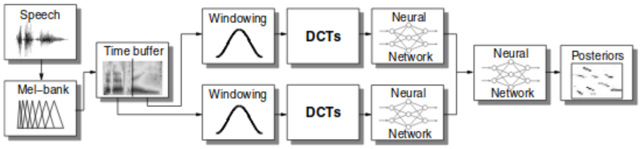


Fig. 2: TRAP-NN model presented in [10]

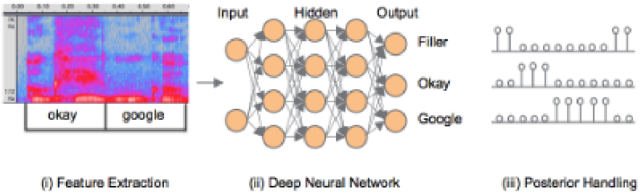


Fig. 3: DNN KWS presented in [12]

C. RNN approach

In [11] a Recurrent Neural Network (RNN) approach is presented, investigating noise contrastive estimation (NCE) and variance regularization (VR). The RNN compress the information from the input layer and computes a new rappresentation, that will be concatenated with the input, using a sigmoid activation function. This is then passed to the output layer to produce normalized RNN probabilities using a sofmax activation. RNN shows better performance than DNN in sequence learning, but when the distance between relevant input signals and the output becomes longer, RNN model loses its learning capability to connect such far information; in order to address this problem, in [13] a Long-short term memory block approach is presented. It is shown that LSTM achieve better results respect to a DNN in KWS task, but unfortunately the authors do not give any information about the structure or the future extraction use for the DNN, except for the number of the layer.

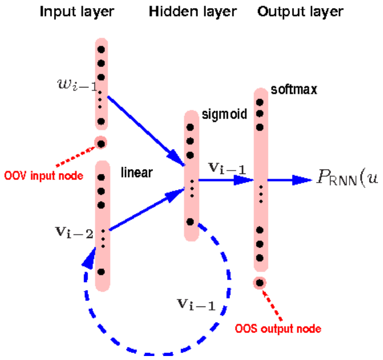


Fig. 4: RNN model presented in [11]

D. CNN approach

In [1] a Convolutional Neural Network for a KWS task is presented. Given an input signal with two dimension time and frequency,a weigth matrix is convolved with full input.The weight matrix spans across a small local time-frequency. This weight-sharing helps model local correlation in the input signal. After performing convolution a max-pooling layer helps to remove variability. CNNs are shown to be better respect to HMM Keyword/Filler model in three different aspects:

- DNNs ignore input topology, while CNNs model exploit local correlation in time and frequency through weights which are shared across local region of input space.
- DNNs are not explicitly designed for model transitional variance within speech signals, which can exist due to different speaking styles. CNNs capture transnational invariance with far fewer parameters by averaging the output of hidden units in different local time and frequency.
- CNNs has higher performance and reduced model size than DNN approaches.

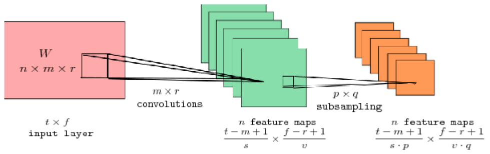


Fig. 5: CNN model presented in [1]

/

III. Processing Pipeline

The project is structured in five basic steps:

- Understanding and analysis of the Tensorflow's speech recognition network. It consists in:
  - Composition and partitioning of the dataset in: training, validation and test dataset.
  - Feature extraction: mell-frequency cepstral coefficients computation.
  - Model architecture definition: two layers convolutional neural network
  - Training, validation and test process.
- Improvements of the Tensorflow's network using different tecniques.
- Design and testing of a more complex architecture: three layer convolutional neural network
- Test of a lighter architecture: one layer convolutional neural network
- Comparison of the architectures and real testing through an android application.

IV. Signals and Features

A. The Dataset

The Dataset consisted of 105,829 utterances of 35 words, broken into categories and frequencies. Each utterance is stored as a one-second (or less) WAVE format file, with the sample data encoded as linear 16-bit single-channel PCM values, at a 16 KHz rate. There are 2,618 speakers recorded. In order to get real audio frame that can work well with cheap devices that are normally used in a noise environment, all utterances were captured through phone or laptop microphones, wherever users happened to be. The database focuses on English, and gathers a variety of accents as wide as possible. All utterances are restricted to a standard duration of 1 second. The dataset used by the model is divided in training set 80%, test set 10% and validation set 10%. Clips are assigned to training test, or validation sets based on a hash of their filename, to ensure that the assignments remain steady even as new clips are added and avoiding any training samples migrating into the other sets. This function is required in order to avoid overfitting: since exist multiple repetition of the same word recorded just by one user, if the model trains on those repetitions and one of them migrates to the test set, the metrics during test could be misleading and inappropriate. The vocabulary is limited, including the digits zero to nine, and ten words that would be useful as commands in IoT or robotics applications; ”Yes”, ”No”,”Up”, ”Down”, ”Left”, ”Right”,”On”, ”Off’, ”Stop”, and ”Go”, plus some others used as a test of model discernment like ”Tree” that have a similar pronunciation to ”Three”. Background and silence where added to the dataset, recording audio from different noisy environment and quiet and irrelevant audio; the number of clips provided with background and silence is set to 10% of the train and test set. Unknown words are added to the dataset, since the application could record word that are neither labeled audio neither noise or silence; in this way a ”unknown” label is trained by the model using word like: ”Bed”, ”Bird”, ”Cat”, ”Dog”, ”Happy”, ”House”, ”Marvin”, ”Sheila”, ”Tree”, and ”Wow”. Control over the number and which unknown word to add into the dataset is provided. In order to enhance the database, a time shifting function is provided; this involves a random offset in time, which defaults to 100ms of the training sample data, so that a small part of the start or end is cut and the opposite section is padded with zeros. Increasing this value will provide more variation, but with the risk of cutting important parts of the audio.

B. Feature Extraction

Audio signal is constantly changing and dealing with non-stationary data makes difficult to extract and label an audio on a few core feature, and this is a serious issue if we want to limit the size of the model and have less parameters, in order to have a real-time audio classification. This is why the one-second WAVE format file, are framed using a 30ms windows, and every clip is sampled at 16 KHz rate, providing 480 sample in total. For each frame a Discrete Fourier Transform (DFT) is applied, converting a time sequence of 480-equally spaced samples into a 480-long complex sequence, which is a complex-valued function of frequency. In framing the signal it is important to consider the overlapping edge: each frame shares the first part with the previous frame and the last part with the next frame; the 10 ms stride of the framing window, indicates how much time there is between the starting time of each frame. Segmenting longer data into shorter finite length DFT inputs does an implicit rectangular windowing, which causes the energy of the frequency of the signals not exactly periodic in the DFT length, to be ”spattered” into other distant frequencies of the DFT result. A non-rectangular window function is applied to reduce these rectangular window artifacts, called spectral leakage [14]. A typical window function, such as Von Hann or Hamming, can be quite lossy at both edges of each window. Overlapping windows adds a window (or more) centered closer to where other windows are lossy, thus helping preserve information about the original signal that would otherwise be lost or degraded. Once DFT is applied to each frame and the spectrogram is provided, it is time to compute Mel-Frequency Cepstral Coefficients, composed by 40 triangular filter, each one multiplied by the entire power spectrum. A filter is centered to a frequency m, calculated on the base of how many triangular filter are needed. When the filter is applied it works like a mask that put to zero the energy trace around the center frequency. Then we take the non-zero resulting coefficient and we apply a logarithm to each one, which tells us how much frequency there is inside each Mel’s frequency band. A Discrete Cosine Transform is then applicate to the vector containing the logarithms of the DCT vector, and the result of the DCT is a vector of 40 cepstral coefficients. A one second audio is windowed in 98 frame, providing 2D fingerprint of 98 frame x 40 DCT coefficient size that is flattened, transforming it to a 3920 features vector.

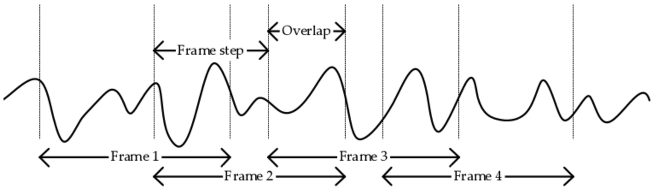


Fig. 6: Audio signal divided in four frame

V. Learning Framework

A. Tensorflow’s model architecture

The original architecture proposed in the Tensorflow ”Simple Audio Recognition” tutorial consists of a convolution neural network (called ”conv”) that takes in input a single channel matrix M t x f ,where t represents the time and f the frequency. The input is fed into a first convolutional layer that consists in 64 filters each ones of dimension 8 x 20. Each filters perform a convolution spanning across the entire surface of the input matrix M, using stride equal to [1,1], which means that when the filter moves it moves in respect of the center of only one position,in the lateral case and in the vertical one. The padding is set as ”SAME” which means that every output will have the same dimension of the input. As activation function is used a ”ReLU function”. The output of the first convolutional layer is given as input to a dropout layer, which in this case discards in a random way the 50% of the input values. Then an operation of max pooling is performed, using a window of 2x2,and stride [2,2], which means that in that window only the highest value is taken, so the output of the max pooling layer will have half the size of the input. A second convolutional layer is used, which consists in 64 filters of dimension 4x10, stride equal to 1, padding set as ”SAME” and ReLU as activation function. Another dropout is performed, with percentage of discard always equal to 50%. The output of the second dropout layer is reshaped in a one-dimension vector to be fed at the last dense layer composed by twelve neurons, that correspond at the number of labels or recognizable words, comprising the ”unknown” and ”silence” labels. This model use as loss function a ”sparse˙softmax˙cross˙entropy” that compute the ”distance” (or error) between the output of the CNN and the corresponding label and convert it into a vector of probability. To perform the operation of backpropagation a classic gradient descent optimizer is used, the larning rate is set at 0.001 for the first 15000 step and at 0.0001 for the last 3000, for a total of 18000 step. At the beginning of this paragraph it has been said that the CNN take as input a single matrix, but in reality a batchsize of 100 is used, so the real input is a set of 100 matrix at time. This kind of ”two convolutional layer” model, as said in the tensorflow tutorial, is pretty large, it takes over 800 million FLOPs for each inference and use 940,000 weight parameters, but any way it can be used on desktop machines or modern phones to perform key word spotting in real time. The problem occure when devices with more limited resources are used and so this kind of architecture could result heavy.A lighter alternative will be presented in the next sections.

B. Improvement of the original network

As said in section Section 2, the first goal of this project is to try to improve the performance of the original neural network but at the same time keep the network relatively light and fast, therefore as first approach we tried to make changes without altering the structure. Various methods have been used:

- **Change activation function (ELU)**: we changed the activation function applied to the two convolutional layers replacing the ReLU function with the eLu activation function, which presents the same shape of the ReLU in positive input range, but differs in the negative ones. The eLu goals are that it should solves the dying ReLU problem and it saturates for large negative value, but not exactly at zero like ReLu.
- **Change the dropout percentage (DRP40)**: dropout is a method used to reduce the probability of overfitting, it acts discarding in a random way a percentage of the input. In this case we reduced the percentage of dropout from 50% to 40% in both the dropout layer. This was done to understand if the percentage of discarded input was to high.
- **Change the batch size (64BS)**: we tried to set the batch size at 64 instead of 100, this wasn’t done to solve problems of memory, but only to decrease the training time, because typically the neural networks trains faster with batches of lower dimension, since the weights are updated after each propagation, so decreasing the batch size, the frequency of back propagation increase.
- **Increase the number of steps(ITH25)**: we increased the number of steps from 18000 to 25000, using 0.001 as learning rate for the first 20000 steps and 0.0001 for the last 5000, this to see if the accuracy would have increased
- **Modified the number of DCT coefficients (DCT15)**: we have decreased the number of DCT coefficients from 40 to 30 an then 15 to decrease the dimension of the input, doing this the network becomes lighter but the number of coefficients may not be enough to have an efficient training.
- **Perform l2 regularization (REG)** : ”weight penalty” regularization also called ”weight decay” is a technique used to prevent the overfitting, its goal is to avoid excessive weight growth keeping them small and to help turn off the irrelevant once. In this case we used an l2-regularization:

$$Costfunction = loss * \lambda * \sum \parallel w \parallel \qquad (1)$$

where λ represents the regularization parameter, in this case set at 0.01.
- **Different gradient descent technique (ADAM)** : Analyzing the structure of the original model, we noticed that the method by which the learning rate was updated could results inefficient, that because in the original code it is changed ”manually” after a certain number of steps (15000). As a consequence of this, we decided to use another type of optimizer, that would solve the problem linked to the non-automatic selection of the learning rate, by choosing the ”Adam optimizer”. Adam stand for ”Adaptive moment estimation” and it is an extension of the classical sthocastic gradient descent, more specifically it is a combination of other two variation of the original one, the ”AdaGrad” and the ”Root Mean Square Propagation”. One of the biggest advantages of the Adam optimizier is that it makes use of the average of the second moments of the gradients to adapt and update automatically the learning rate value.Specifically, the algorithm calculates an exponential moving average of the gradient and the squared gradient, and it has two parameters $\beta 1$ and $\beta 2$ that control the decay rates of these moving averages. Nowadays Adam is one of the most popular algorithm in the field of deep learning because it achieves good results in a short time, how we will se in the next paragraph.
- **Combinatons of the previous methods(AD25*)**: After testing all the previous methods, we have selected those that produced good results and we tried to combine them to achieve some further improvements. Firstly we tried to increase the number of step up to 25000 and use the Adam optimizer instead of classic gradient descent optimizer in the original model, afterwards, we have added to the previous structure the l2 regularization.Starting from this last combination we have created a new model which comprises a second max pooling layer positioned at the

end of the second convolutional layer, which has the same feature of the first one, so a window size of 2 x 2 and a stride of [2,2],in this way the number of parameter decrease. Always using the combination of Adam, 25000 steps and l2 regularization we tried to change the dimension of the filters of the two convolutional layer. In the first test we used 64 filters of dimension 5 x 5 in the first convolutional layer and 128 filters of dimension 5 x 5 in the second one. Instead in the second test we used 64 filters of dimension 20 x 10 in the first convolutional layer and 64 filters of dimension 5 x 5 in the second one.

C. *Three convolutional layer architecture(AD25R3L)*

Once we have improved the performance of the model with two convolutional layer we focusing on increasing the network performances regardless of its efficiency in terms of time and complexity.We have decided to create a model with three convolutional layer. Initially we started from the original model and at the end of the second convolutional layer we attached a max-pooling layer with window size of 2 x 2 and stride of [2,2] and then the third convolutional layer composed by 128 filters of dimension 4 x 4, always with ReLu as activation function, that it will be reconnected at the dense layer of the original model.This model has been trained for 25000 step. Afterwards we improved the network adding l2 regularization and replacing the gradient descent optimizer with Adam.

D. *One convolutional layer architecture(AD1L)*

In the tensorflow's tutorial is proposed a lighter neural network model as alternative to the two convolutional layer model, this because the last mentioned could involves too many calculations to run at interactive speeds on devices with more limited resources. In terms of number of weight parameters, the one convolutional layer model is equivalent to that of two, but in terms of FLOPs it need only 11 million to do a prediction instead of 800 million. This lighter version called "low latency conv model" consists in a neural network composed by: a single convolutional layer of 186 filters of dimension 8 x input˙time˙size, stride of [1,1], with ReLu as activation function, two dense layer of 128 neurons each and a final dense layer of twelve neurons, again corresponding at the number of labels. This model train for 20000 steps with 0.01 as learning rate and other 6000 steps at 0.001, in total 26000 steps. We tested a second version of this lighter model replacing the classic sthocastc gradient descent with the Adam optimizer.

E. *Real test with android application*

Tensorflow Speech Tutorial come with four Android application examples:TF Classify, TF Detect, TF Stylize, TF Speech. TF Speech examples runs a simple speech recognition model built using the speech tutorial. Listens for a small set of words, and highlights them in the UI when they are recognized. We substitute the original model with our models,evaluating the KWS performance.

VI. Results

Many algorithms have been proposed in the past for KWS implementation, and several techniques have been described for being very efficient and accurate. However, when it comes to compare the expected behavior of a technique for a new keywords and utterances,the generalization of published comparisons is not very clear, and the choice of the benchmark-keywords has considerable effects on the comparison [15]. In the following Table we have listed the models described above, with the respective number of parameters (P), validation accuracy (V.A), Test Accuracy (T.A), Steps (S.) and training time (t).

Model	P	V.A	T.A	S	t
ORI	926K	88.5 %	87.5	18K	2.43 h
ELU	926K	86.1%	84.7%	18K	2.38 h
64BS	926K	86.1%	84.7%	18K	2.38 h
ADAM	2M	91.7%	91.0%	18K	2.40 h
ITH25	926K	90.04%	88.3%	25K	3.47 h
REG	926K	89.0%	87.5%	18K	2.50 h
DCT15	475K	88.9%	87.2%	18K	2.22 h
DCT30	738K	88.2%	87.2%	18K	2.44 h
DRP40	926K	88.1%	86.8%	18K	3.05 h
AD25	2M	88.1%	86.8%	25K	3.05 h
AD25R	2M	92.8%	92.4%	25K	3.50 h
AD25R2M	1M	92.6%	91.5%	25K	3.40 h
AD25RA1	4M	91.4%	90.2%	25K	3.50 h
AD25RA2	2M	92.8%	92.0%	25K	3.50 h

As we can evince, looking at the model ITH25 augmenting the number of steps to the original model (ORI) improve accuracy. Also the number of DCT coefficients can be smaller, since model DCT15 does not get worse accuracy performances moreover decrease the number of parameters used by the model, saving time and computation resources. ADAM model refer to the ADAM method that achieve high level of accuracy in a short-time range; taking a closer look to the fig.7, we can notice that with 2000 step the model accuracy is around 80%. The AD25R2M outperform all other models,92.6% of validation accuracy and 91.51% test accuracy. Also fig.10 show that AD25R2M is more likely to be diagonal than the original model that shows higher misclassification fig. 11. AD25R2M has less parameters respect to the other models that achieve an accuracy around 90%.

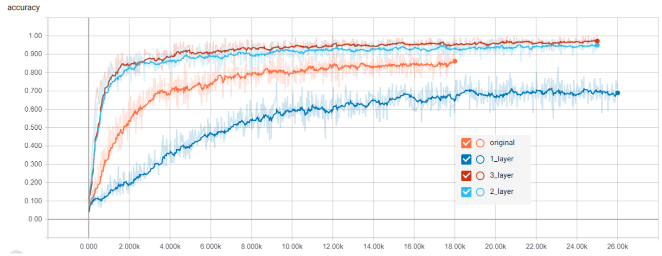


Fig. 7: ORI,AD25R2M , AD25R3L and AD1L models training accuracy from Tensorboard

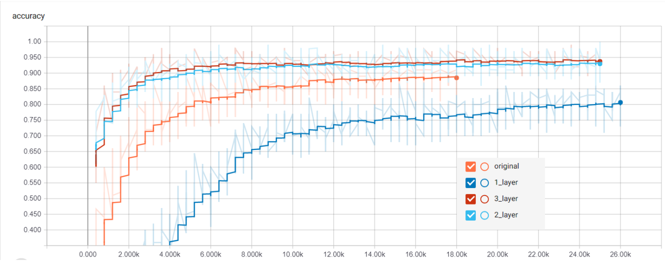


Fig. 8: ORI,AD25R2M , AD25R3L and AD1L models validation accuracy from Tensorboard

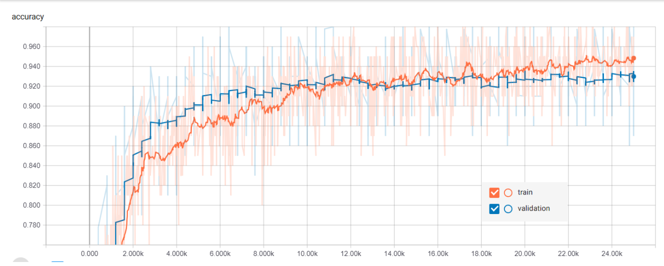
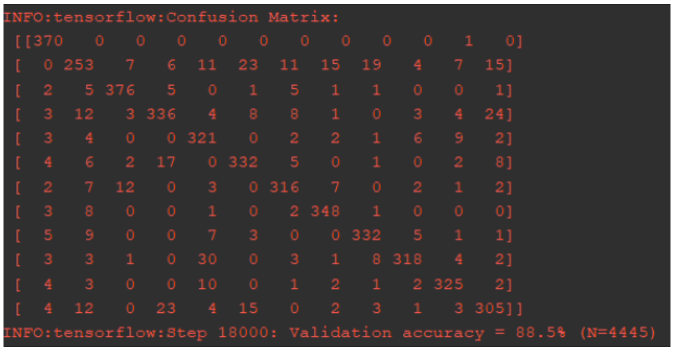
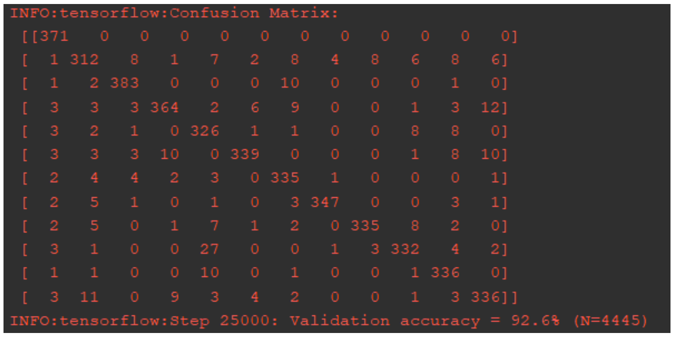


Fig. 9: AD25R2M model training accuracy from Tensorboard



```
INFO:tensorflow:Confusion Matrix:
 [[370   0   0   0   0   0   0   0   0   0   1   0]
 [  0 253   7   6  11  23  11  15  19   4   7  15]
 [  2   5 376   5   0   1   5   1   1   0   0   1]
 [  3  12   3 336   4   8   8   1   0   3   4  24]
 [  3   4   0   0 321   0   2   2   1   6   9   2]
 [  4   6   2  17   0 332   5   0   1   0   2   8]
 [  2   7  12   0   3   0 316   7   0   2   1   2]
 [  3   8   0   0   1   0   2 348   1   0   0   0]
 [  5   9   0   0   7   3   0   0 332   5   1   1]
 [  3   3   1   0  30   0   3   1   8 318   4   2]
 [  4   3   0   0  10   0   1   2   1   2 325   2]
 [  4  12   0  23   4  15   0   2   3   1   3 305]]
INFO:tensorflow:Step 18000: Validation accuracy = 88.5% (N=4445)
```

Fig. 10: Original confusion matrix model’s



```
INFO:tensorflow:Confusion Matrix:
 [[371   0   0   0   0   0   0   0   0   0   0   0]
 [  1 312   8   1   7   2   8   4   8   6   8   6]
 [  1   2 383   0   0   0  10   0   0   0   1   0]
 [  3   3   3 364   2   6   9   0   0   1   3  12]
 [  3   2   1   0 326   1   1   0   0   8   8   0]
 [  3   3   3  10   0 339   0   0   0   1   8  10]
 [  2   4   4   2   3   0 335   1   0   0   0   1]
 [  2   5   1   0   1   0   3 347   0   0   3   1]
 [  2   5   0   1   7   1   2   0 335   8   2   0]
 [  3   1   0   0  27   0   0   1   3 332   4   2]
 [  1   1   0   0  10   0   1   0   0   1 336   0]
 [  3  11   0   9   3   4   2   0   0   1   3 336]]
INFO:tensorflow:Step 25000: Validation accuracy = 92.6% (N=4445)
```

Fig. 11: AD25R2M confusion matrix model’s

Model	P	A.V	A.T	S	t
ORI3L25	689K	89.4%	87.9%	25k	3.50 h
AD25R3L	2M	93.5 %	93.7%	25k	3.50 h
1L	949K	79.4 %	77.6%	26k	2.50 h
AD1L	2M	80.0 %	78.0%	25k	2.50 h

The Table above lists the models with the respective number of parameters,confirming that adding a layer to the AD25R2M, resulting in AD25R3L, do not improve significantly the accuracy to deferment of the number of parameters.

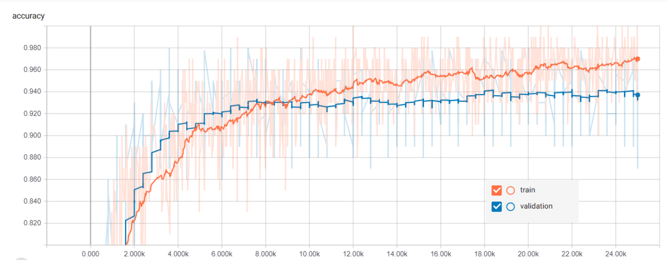


Fig. 12: AD25R3L training and validation accuracy from Tensorboard

Moreover the ADAM method used in the AD1L model is not efficient because,as the fig.13 point out,there is too much variability between training and validation values.

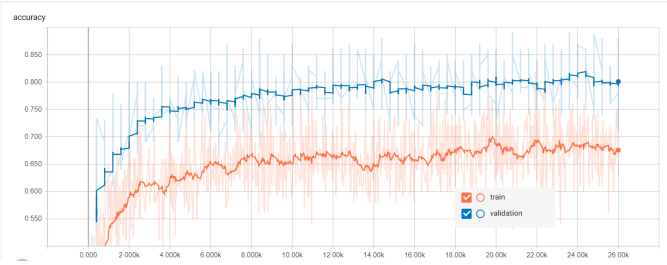


Fig. 13: AD1L training and validation accuracy from Tensorboard

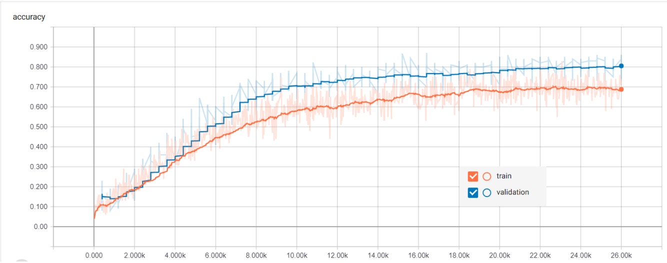


Fig. 14: AD1L training and validation accuracy from Tensorboard

We use the Android source code to build and compile two different applications, one using AD25R2M model and the other using AD1L model, The application run over a Nexus 4 Android 4.2,mounting a chipset Qualcomm APQ8064 Snapdragon S4 Pro with a CPU Quad-core 1.5 GHz Krait and Adreno 320, using 2 GB RAM. The test were performed in a silent and a noisy environment, we asked to 2 male and 2 female subject to test the app, asking them to say a list of 20 random word,composed by label word and unknown word used during the training phase. From the resulted list of word classified by the model we measure:

- Word Correct Rate: the percentage of word labeled correctly with respect to the total words presented.
- Word Error Rate: the percentage of words erroneously classified with respect to all words presented.
- Levenshtein distance: is a string metric for measuring the difference between two sequences. The Levenshtein distance between two words is the minimum number of single-character edits, insertions, deletions or substitutions, required to change one word into the other.
- Precision: is the fraction of relevant instances among the retrieved instances.

$$Precision = (Corr - Ins)/Corr + Sub + Ins \tag{2}$$

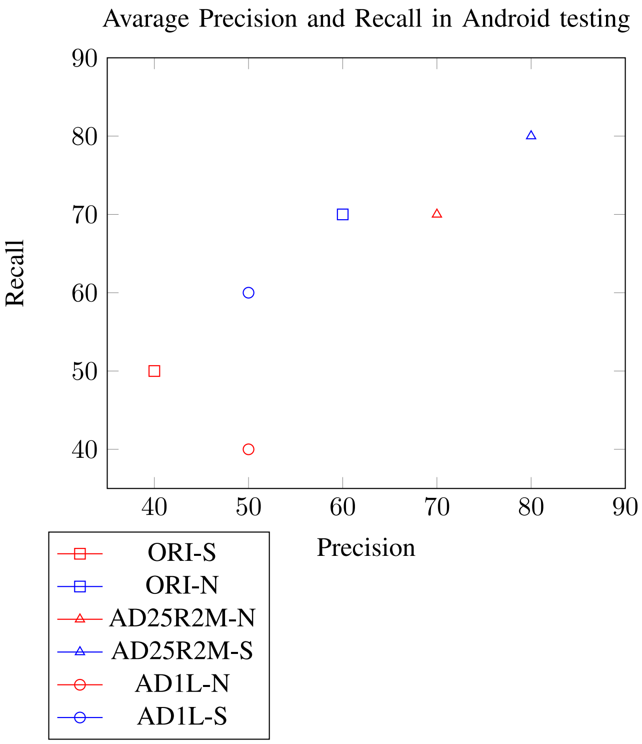
- Recall: is the fraction of relevant instances that have been retrieved over the total amount of relevant instances.

$$Recall = Corr/(Corr + Sub + Del) \tag{3}$$

The following Table,lists the average Word Error Rate and the Word Correct Rate per model.

Model	WCR	WER
ORI-N	50%	50%
ORI-S	65%	25%
AD25R2M-N	65%	35%
AD25R2M-S	75%	15%
AD1L-S	55%	45%
AD1L-N	50%	50%

All models perform better in a silent environment, AD25R2M-N perform better than the other models, but what this result point out more is that ORI-N and AD1L-N have achieve the same result in noisy environment, showing that augmenting the layer without tune better the simpler model parameters do not provide better results. The graph summarize the result, averaging precision and recall for each model, that we achieve during user testing, showing that AD25R2M outperform in both silent and noisy environment, confirming what the accuracy table and graph previously already state, passing from 928K parameters in the original model to 1M in AD25R2M model. Meanwhile the AD1L has the lower performances, but still give us interesting result, taking into account that this structure have less parameters and require less resource to compute a classification. Levenshtein distance show that couple of words like, ”Dog” and ”On” or ”Cat” and ”Yes”, that are near, have higher probability to be misclassified.



VII. Concluding Remarks

Analyzing the whole project, we started from the ”Simple Audio Recognition” tensorflow tutorial with the basic idea that the proposed model could be improved in terms of performance, and as demonstrated, after analyzing the architecture and identifying the weak points , by applying some methods the goal has been achieved. We have seen that only by increasing the number of steps the validation and testing accuracy has increased from 88.5%/87.5% to 90.5%/88.3%. A fundamental role was played by the ”Adam optimizer” that combined with other methods increased the accuracy at 92%-93%, to the detriment of a grouth of the number of the parameters. Another good result achieved thanks to adam, is the fact that is possible to obtain a relatively high accuracy using many fewer steps, with the AD25R2M model with 4000 steps it reach the 83% instead of 53%, that could be useful using using machines with low performance. Talking about the number of parameters, we have seen that max pooling or in this case the decrease of the number of dct coeffcients can be useful to make the model lighter and maintain the accuracy percentage. In relation to the one and three convolutional layer models, they have been proposed to offer alternatives to be used as needed. Talking about the AD25R3L it reach the highest accuracy both in validation and test compared with the other model, but the improvement is relatively small compared to the increase in complexity of the model, so it doesn’t result so efficient. In the case of the one convolutional layer model we have shown the adam optimizer not always improve the result. In the paper it was seen that to build a good system of speach recognition, using deep learning, it is essential to take into account of the improvement of the accuracy,to make the system more performing, but at the same time keep under controll the number of weight parameters and the FLOPs, so in conclusion, a possible development of the project could be to develop a lighter model that maintains or even increases the accuracy. This goal could be achieved by using or combining other methods or planning othe structures with a different schema. This would allow to use the system even on less performing platforms or simply to overcome the problems of overheating or slowing down the devices.

VIII. Moreover

Having already had some past experiences in building convolutional networks from scratch, in this project we focused more on managing and improving an already pre-set neural network, learning to use various methods for develop and modify a network. The main difficulties were found in understanding how the system worked, considering that there are different libraries and methods to develop a CNN,and which changes to use to improve the performance.A big problem that characterizes machine learning in general is that training takes some time to complete, so the number of tests is limited.

References

[1] T. Sainath and C. Parada, "Convolutional Neural Networks for Small-footprint Keyword Spotting," in *INTERSPEECH 2015*, (Google, Inc. New York, NY, USA), 2015.

[2] V.B.Len, L. Besacier, and T. Schultz, "Acoustic- phonetic unit similarities for context dependant acoustic model portability," in *Carnegie Mellon University*, (Pittsburgh,PA, USA, year = 2006).

[3] E. Singer, P. A. Torres-carrasquillo, T. P. Gleason, W. M. Campbell, and D. A. Reynolds, "Acoustic, phonetic, and discriminative approaches to automatic language identification*," in *International Journal of Computer Applications*, 2010.

[4] D.R.Reddy, "An Approach to Computer speech Recognition by direct analysis of the speech wave," in *Tech.Report No.C549*, (Computer Science Department,Stanford University,sept.1996), 1996.

[5] T. R.K.Moore, "Twenty things we still do not know about speech processing," in *CRIM/FORWISS Workshop on Progress and Prospects of speech Research and Technology 1994*, 1994.

[6] K. S. et.al., "Speech Recognition using weighted HMM and subspace," in *IEEE Transactions on Audio, Speech and Language*, Mar. 1994.

[7] S. et.al., "A New hybrid algorithm for speech recognition based on HMM segmentation and learning Vector quantization," in *IEEE Transactions on Audio Speech and Language processing Vol.1,No.4*.

[8] M.J.F.Gales and S. young, "Parallel Model combination for Speech Recognition in Noise technical Report," 1993.

[9] A.P.Varga and R.K.Moore, "Hidden Markov Model Decomposition of Speech and Noise," in *Proc.ICASSp, pp.845-848*.

[10] I.Szoke, P. Schwarz, and P.Matejka, "Comparison of Keyword Spotting Approaches for Informal Continuous ," (Boulder, CO, US).

[11] J. V. X. L. K. M. G. X.Chen, A. Ragni, "Recurrent neural network language models for keyword search," *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5775–5779, 2017.

[12] G. Chen, C. Parada, and G. Heigold, "Small-Footprint Keyword Spotting Using Deep Neural Networks," in *Center for Language Speech Processing*, (J.H. University,Baltimore,MD - Goolge Inc., Mountain View,CA), 2014.

[13] J. S. J.ZHANG, L.HUANG, "Keyword spotting with long short-term memory neural network architectures," *2017 International Conference on Computer, Electronics and Communication Engineering (CECE 2017)*, 2017.

[14] R. Elder, "Endpoint discontinuities and spectral leakage," *Robert Elder software inc.*, vol. http://blog.robertelder.org/endpoint-discontinuities-spectral-leakage/, Feb. 2018.

[15] M. C. Silaghi, "A new evaluation criteria for keyword spotting techniques and a new algorithm," in *In: Proceedings of InterSpeech. (2005*, 2005.