**Morpheus Team 2016**

**Year 1**

Technical Report

Morpheus Team

**Document Changes**

|  |  |  |  |
| --- | --- | --- | --- |
| # | Description | Author | Date |
| 1 | Adding initial structure for MFCC investigations | Thomas Paula | 07/26/16 |
| 2 | Adding introduction and information regarding libraries | Thomas Paula | 07/28/16 |
| 3 | Adding information regarding datasets and Bob library | Wagner Rampon | 07/28/16 |
| 4 | Explaining how MFCC steps work | Rafael Borges | 07/28/16 |
| 5 | Finishing report structure to Sprint Review | Thomas Paula/Wagner Rampon | 07/29/16 |
| 6 | Adding initial structure for audio split and speaker diarization investigation, adding libraries information, adding naive approach information to audio splitting | Wagner Rampon | 08/10/16 |
| 7 | Continuation of item 6 this section | Gabrielle Marchioro/ Rafael Borges | 08/11/16 |
| 8 | Restructured the report | Thomas Paula | 08/24/16 |
| 9 | Adding GMM and UBM information | Wagner Rampon | 10/24/16 |
| 10 | Adding CNNs information | Thomas Paula | 10/24/16 |

**Table of Contents**

[Speaker Diarization](#_g1pu7ih9jnjz)

[Introduction](#_ap4z0rlxectt)

[1. MFCC - Mel-Frequency Cepstral Coefficients](#_7x2ni45dgn49)

[1.1. Introduction](#_wxp9c5n8txo8)

[1.2. How it works](#_bddgv1mhjn65)

[1.2.1. Preemphasis](#_40ahqfwx5bvc)

[1.2.2. Windowing](#_kzijphhcpyt9)

[1.2.3. Fast Fourier Transform (FFT)](#_k1xedar8ydu2)

[1.2.4. Power](#_cwzj4xhxeqz)

[1.2.5. Mel Frequency Warping](#_kl3mhv16g1db)

[1.2.6. Log Compression](#_ugf2pnnrt3yo)

[1.2.7. Discrete Cosine Transform](#_apuxb55lvzg4)

[1.2.8. Smoothing](#_p7cegcb9vgql)

[1.2.9. Liftering](#_k6zhn7aa7nl4)

[1.3. Libraries](#_km7izdh7p4l)

[1.3.1. Python Speech Features](#_yv53niz5u7rd)

[1.3.2. Librosa](#_45m314w50mt8)

[1.3.3. Scikits Talkbox](#_id20hrfcipzt)

[1.3.4. Bob Bio](#_kdpmlkp0p3pq)

[1.4. Datasets](#_av93z3zaq40a)

[1.4.1. VoxForge](#_t9s4r0ys4e9y)

[1.4.2. MOCHA TIMIT](#_4co3w6m4gutj)

[1.4.3. Dataset Distribution Portal - Youtube](#_t1udqa9kinlb)

[2. Audio Split](#_o9pxrgxgci55)

[2.1. Introduction](#_lsva9ghicn1w)

[2.1.1. Audio Split](#_7nwa4koysfdl)

[2.2. Libraries](#_ggirbnt6urjm)

[2.2.1. PYCASP - Python-based Content Analysis Using Specialization](#_9xd82sdjx66b)

[2.2.2 PyAudioAnalysis - A Python Audio Analysis Library](#_hb9hm8i1hv61)

[3. Gaussian Mixture Models (GMM)](#_svmovoaza30m)

[3.1. Introduction](#_qh8cwqm5m7fg)

[3.2. Expectation Maximization](#_ow9536s1waf9)

[4. Universal Background Model (UBM)](#_bol8u0hylk0j)

[4.1. Introduction](#_2vgmr1p32mjy)

[4.2. Adaptation of Speaker Model](#_sp603kr1yvbj)

[5. Audio Analysis Library](#_e5yhx4x0w6l6)

[TODO](#_swdl1n5tlzil)

[6. Deep Learning for Audio Feature Extraction](#_nky3jhirv20d)

[6.1. Introduction](#_za8zazm6kv1c)

[6.2. Papers Evaluated](#_dwv0cn1c0noe)

[6.2.1. WaveNet [6]](#_ognj7vhug20h)

[6.2.2. Estimating phoneme class conditional probabilities from raw speech signal using convolutional neural networks [7]](#_e12y51adgd1u)

[6.2.3. Acoustic Modeling with Deep Neural Networks Using Raw Time Signal for LVCSR [8]](#_s99m9mit8qvq)

[6.2.4. Convolutional, Long Short-Term Memory, Fully Connected Deep Neural Networks [9]](#_2bmc8ruxsoj0)

[6.2.5. Learning the Speech Front-end With Raw Waveform CLDNNs [10]](#_n53v208yrro8)

[6.2.6. Speech Acoustic Modeling From Raw Multichannel Waveforms [11]](#_1a2pyiuppixv)

[6.2.7. Conclusion](#_6ig5egas4l5r)

[Bibliography](#_xajyxuac0si1)

# 

# 

# **Speaker** Diarization

Identifying who spoke when in a meeting environment

## 

## 

## Introduction

Speaker recognition can be understood as the identification of a person through the voice and its characteristics. It is divided in speaker identification and speaker verification. Speaker identification aims to identify who is speaking given an audio sample. It identifies the speaker among a set of known speakers. On the other hand, speaker verification checks whether the voice is of a specific person or not. There is also an area within Speaker Recognition that is called Speaker Diarization. It aims to is to annotate temporal regions of audio recordings with speaker labels, in order to answer the question “who spoke when”.

All the three problems are comprised of many different challenges within. For instance, extracting features from raw audio and classifying it is a very challenging task. Besides, capturing audio *per se*, filtering and generating audio files is also a cumbersome. To tackle such a problem, our approach was to first understand the classical implementations, which are mostly based on Mel-Frequency Cepstral Coefficients (MFCC).

MFCC based approaches usually extract features from raw audio with MFCC and use such features as input to a classifier (Logistic Regression, K-NN, Neural Network, etc). There are also approaches that are based on Gaussian Mixture Models (GMMs), which are widely used in both academia and industry. Recently, Deep Learning approaches achieved even better results, being able to extract relevant features from raw audio. Another challenge is to find suitable datasets to train the algorithms on. We were able to find an interesting quantity of datasets, which are described in details in experiments sessions of each subsession.

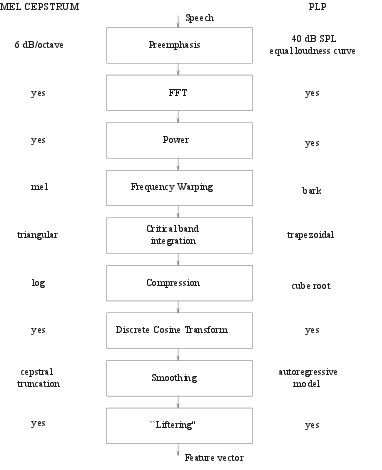
This document describes all investigations and experiments made with respect to speaker recognition, focusing on the speaker diarization.

## 1. MFCC - Mel-Frequency Cepstral Coefficients

### 1.1. Introduction

The main idea of MFCC consists in getting the prevalence of certain frequencies in a given audio signal. It is used for speaker recognition based on the hypothesis that each speaker have a different frequency signature for his voice. The choice of frequencies is based on the Mel scale - a perceptual scale of pitches judged by listeners to be equal in distance from one another[1]. After getting the components for each frequency, the cosine transform is applied to the components taken as if they were a new spectrum, in order to get the power cepstrum components.

As cited in [2], the basic blocks for MFCC implementation are as follows:



### 

### 

### 1.2. How it works

#### 1.2.1. Preemphasis

Given the difference of the signal amplitude on different frequencies for the human voice, a pre-emphasis step is used in order to flatten the spectrum. Usually, low frequencies have a higher amplitude, so a high-pass filter is often used. Two simple implementations of pre-emphasis are the subtractive and the additive.

#### 1.2.2. Windowing

A given signal is actually represented as an array of equally spaced samples of the amplitude of this signal. The rate which the signal is sampled is usually called frame rate. In order to apply the transforms, small chunks of the signal need to be provided. Usually, these small chunks (windows) have about 25 milliseconds, using a step distance of 10 ms between windows. The size of each window is then defined by the product of frame rate(in samples per second) by size of the windows (in second).

#### 1.2.3. Fast Fourier Transform (FFT)

The fourier transform converts a window of the signal into the coefficients for a different range of frequencies. For a window with a size *t*, the algorithm captures (t / 2) + 1 components, where the first gives the energy gain, and all the others give the information about a specific frequency (1 cycle/t, 2 cycles/t, ... t/2 cycles/t). This coefficient is actually a complex number, but a simple implementation usually strips off the imaginary part.

#### 1.2.4. Power

The coefficient for each frequency is then squared (usually), in order to change the scale into a more suitable for audio purposes.

#### 1.2.5. Mel Frequency Warping

The distance between two consecutive frequencies given by the FFT, is regarding the Hertz space, i.e. number of cycles per time unit. In order to allow a decomposition which is more suitable for audio purposes, the frequencies are then warped to the mel scale, in which the distance between consecutive frequencies is more related to the distance between two sounds as perceived by a human being.

In order to do so, the lower and higher frequency of the spectrum are converted into a MEL value. The new Mel space is divided equally to obtain n base frequencies. These frequencies are then converted back to the Hertz representation. The input signal is then submitted to triangular filters, centered in the n defined base frequencies.

#### 1.2.6. Log Compression

The coefficients are then submitted to a logarithmic function, in order to provide a new change of the scale, also suitable for audio domains.

#### 1.2.7. Discrete Cosine Transform

As an approximation of the power cepstrum for each of the given frequencies, the Discrete Cosine Transform is applied to the array of frequencies as if they were a new signal. This will return a set of coefficients that represent the different levels of the signal.

#### 1.2.8. Smoothing

The process of smoothing is extremely simple: the array of coefficients is truncated, eliminating the last values. This is made in order to discard values which represent properties of little relevance for audio processing purposes.

#### 1.2.9. Liftering

The liftering step is the last application of a trigonometric function in order to (possibly) enhance the quality of the cepstral information for processing purposes.

### 1.3. Libraries

We explored some libraries for extracting MFCC, focusing on Python implementations. In the next sections we detail which one of them and our main concerns about each one.

#### 1.3.1. Python Speech Features

**Website**

<https://github.com/jameslyons/python_speech_features>

**Installation**

It can be installed simply using pip:

$ pip install python\_speech\_features

**How it works**

Python Speech Features contains methods to extract common features for Automatic Speech Recognition (ASR), such as MFCCs and filterbank energies. Each of the methods contain a variety of parameters that enable one to change window sizes and sample rate, for instance. The default parameters used in method to extract MFCC in this library returns 13 coefficients. There is also a method to extract the Mel-filterbank energy features from the audio signal, which returns 26 coefficients from the audio signal.

**Experiments**

We collected audio from two different people speaking from YouTube. One audio is from Hillary Clinton whereas the other is from Donald Trump. We loaded the audio regularly using Scipy’s Wav package and extracted the MFCC and the Log Filterbank from the audio. We selected one of the windows (5000) and plotted the values. The blue line is from Hillary Clinton while the green is from Donald Trump. We can clearly see that there are some differences on the values, which indicates that the extracted features could be used to differentiate them.

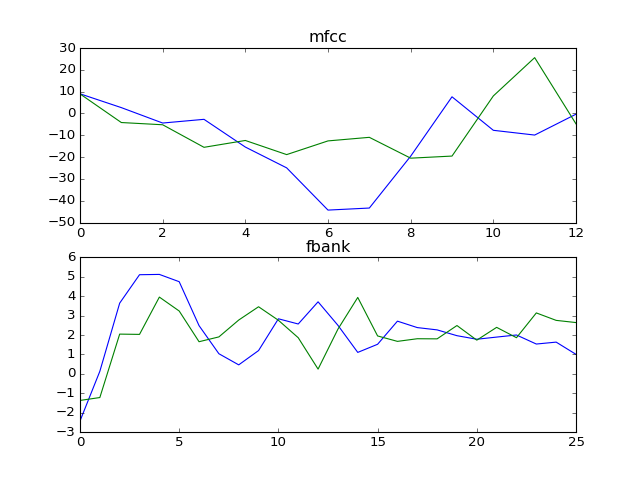


Figure shows the results of features extracted with Python Speech Features for two different people.

A Second experiment consisted of extracting and plotting the mfcc obtained from two sound files from the same speaker, saying the same text but with a separation of one octave (twice the frequency).



Figure shows the overlap of the extracted features with Python Speech Features from the two audio sources along each frame.

#### 1.3.2. Librosa

**Website**

<https://github.com/librosa/librosa>

**Installation**

It can be installed simply using pip:

$ pip install librosa

**How it works**

Librosa is one of the most used Python libraries for extracting features from audio. It contains methods to load the audio, compute features and extract and plot spectrograms. It is also integrated with IPython/Jupyter notebooks, enabling one to create a player and play audio directly in the notebook. Its default way to extract MFCC from an audio yields 13 coefficients from the audio signal.

**Experiments**

We used the same audios described in Python Speech Features experiments. One audio is from Hillary Clinton whereas the other is from Donald Trump. We loaded the audio using Librosa methods and extracted the MFCC of the audio. We selected one of the windows (5000) and plotted the values. The blue line is from Hillary Clinton while the green is from Donald Trump.

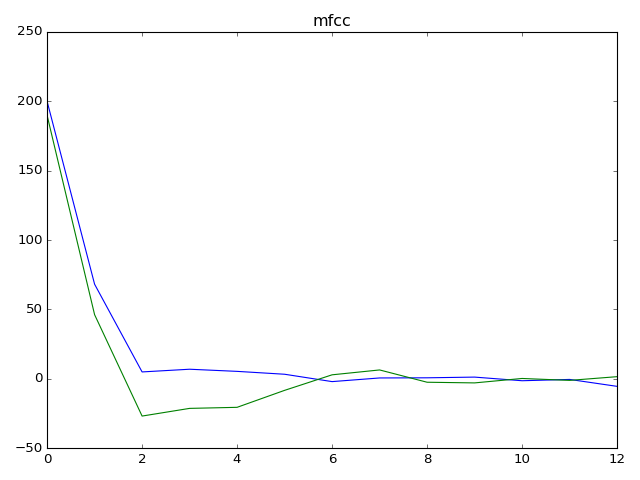


Figure shows the results of MFCC features extracted with Librosa for two different people.

#### 1.3.3. Scikits Talkbox

**Website**

<https://github.com/librosa/librosa>

**Installation**

It can be installed simply using pip:

$ pip install scikits.talkbox

**How it works**

Scikits Talkbox is a set of python modules for speech and signal processing. It is currently something separated from the original Scipy but that could be integrated on the main libraries eventually. Librosa is one of the most used Python libraries for extracting features from audio. It contains methods compute features from the audio. Its default way to extract features is able to extract three different ones, namely ceps (Mel-cepstrum coefficients), mspec (Log-spectrum in the mel-domain) and spec (no documentation available). The first one yields 13 coefficients, whereas the second yields 40 and the last one 512.

**Experiments**

We used the same audios described in Python Speech Features and Librosa experiments. One audio is from Hillary Clinton whereas the other is from Donald Trump. We loaded the audio using scipy methods and extracted the three before-mentioned features of the audio. There are three plots, each of which showing information of one of the three features.

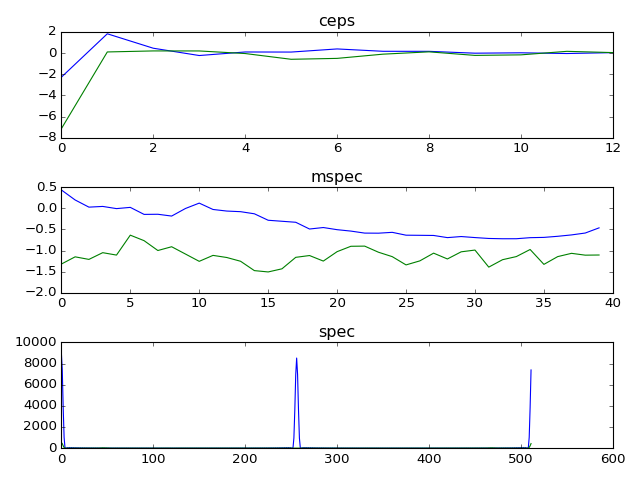


Figure shows the results of extracted features using Scikit Talkbox for two different people.

#### 1.3.4. Bob Bio

[Bob](https://github.com/idiap/bob/wiki) is a free signal-processing and machine learning toolbox originally developed by the Biometrics group at Idiap Research Institute, in Switzerland. Bob has several modules, including [speaker identification](https://pythonhosted.org/bob.bio.spear/).

Despite being written in python, installation is not trivial and there are several dependencies (both on the OS packages and sequential steps to setup the library). The amount of parameters, and lack of proper documentation makes the setup process and the usage of the library with custom datasets a difficult task, requiring the developer to open multiple source code files to understand what are the expected inputs, dataset organization and even processing of the results after applying the training algorithms. Furthermore, there are obscure errors and inconsistent results during both installation and runtime that make the execution success completely unpredictable.

### 1.4. Datasets

#### 1.4.1. VoxForge

[Voxforge](http://www.voxforge.org/) offers a collection of transcribed speech for use with Free and Open Source Speech Recognition Engines. A small subset of the english audio files (only 6561 files) belonging to 30 speakers randomly selected was downloaded to be used on our experiments.

#### 1.4.2. MOCHA TIMIT

A set of 460 sentences designed to include the main connected speech processes in English (eg. assimilations, weak forms). Sentences spoken by 3 different speakers (2 male and 1 female) with different accents of English. All recordings were made in the same sound damped studio at the Edinburgh Speech Production Facility. All data were recorded direct to computer and carefully synchronised.

#### 1.4.3. Dataset Distribution Portal - Youtube

[The YouTube personality dataset](https://www.idiap.ch/dataset/youtube-personality) consists of a collection of behavioral features, speech transcriptions, and personality impression scores for a set of 404 YouTube vloggers that explicitly show themselves in front of the a webcam talking about a variety of topics including personal issues, politics, movies, books, etc. There is no content-related restriction and the language used in the videos is natural and diverse.

### 

## 2. Audio Split

### 2.1. Introduction

#### 2.1.1. Audio Split

There are naive approaches for audio split that do not consider the continuum of a speaker nor interruptions. One of those approaches is the usage of a noise gate which acts as a high-pass filter that does not consider waves below a certain amplitude. Therefore, each time a region of audio was isolated between those lower amplitude segments, a slice would be made.

There are several problems with approaches like these, mainly due to background voices not being considered, signal to noise ratio, estimating the parameter in which the gate considers or not a sample as voice candidate, or spikes of audio due to other sources being sliced as voice candidates.

### 

### 

### 2.2. Libraries

#### 2.2.1. PYCASP - Python-based Content Analysis Using Specialization

The [library](https://github.com/egonina/pycasp/wiki) aims to provide a single software environment for productive, efficient, portable and scalable application development. PyCASP is a collection of specializers (mini-compilers) that automatically map computations onto parallel processors (NVIDIA GPUs, Intel multicore CPUS and clusters). PyCASP targets audio content analysis applications (speech and music processing for example) however, the specializers can be used for other applications. One of the supported features present on the library is of Speaker Diarization. That is, is to determine "who spoke when?" in an audio recording. The algorithm is based on agglomerative hierarchical clustering of GMMs using the Bayesian Information Criterion (BIC) to segment the audio feature files into speaker-homogeneous regions.

An attempt was made to install the library both on Windows, Ubuntu LTS 16.04 and Ubuntu LTS 14.04, all which were unsuccessful. Several prerequisites of the library were discontinued, made unavailable, or poorly documented. For instance, one of the pre-requisites boost deeply modified its structure, so a search was necessary to find a compatible older version coinciding with PYCASP release date (july, 2013).

For all those reasons, and also because the library utilizes approaches deemed obsolete by current standards, along with time constraints, no further investigations were made.

#### 2.2.2 PyAudioAnalysis - A Python Audio Analysis Library

It is an open Python [library](https://github.com/egonina/pycasp/wiki) that is covering a wide range of audio-related functionalities focusing on feature extraction, classification, segmentation and visualization issues. The library seems to be always updated , the last update was 08/10/16.

An attempt was made to install the library on Windows and Ubuntu LTS 14.04, the first one was unsuccessful. A large number of prerequisites was necessary to install the library on Windows, and the installation became complicated.

The actual work is based in two functionalities that is segmentation and diarization. The main idea is to test the audio from AMI dataset and analyse the datas from there applying these functionalities.

## 3. Gaussian Mixture Models (GMM)

### 3.1. Introduction

Gaussian Mixture Models is a parametric probabilistic function which can be used to represent random variables whose distribution are not known. Several diarization solutions apply gaussian mixture models on some step of the process, mostly to obtain new set of features.

Gaussian Mixture Models rely on gaussian distributions. Each observation has different degrees of probability of belonging to a certain mixture. Another way of putting it is to say that each mixture has a given responsibility for an observation. Therefore, mixture models have a certain soft-labeling behavior. Since the data is not always labeled, the parameters for the mixture components (mean and variance) have to be estimated according to the available data. This is performed through an algorithm called Expectation Maximization.

### 3.2. Expectation Maximization

The Expectation Maximization is an iterative method for finding maximum likelihood (or, the local minima) for a set of unlabeled observed variables. As the name states, the algorithm has two steps:

* Expectation - In this step, the responsibility that mixtures have over each observation is calculated.
* Maximization - Here, the parameters of each mixture is updated considering the calculated responsibility on the previous step.

These steps are alternated until a convergence is achieved. There are several metrics to evaluate convergence, and one of the most popular is the logarithmic likelihood. A presentation with more detailed explanations is available on the project drive.

## 4. Universal Background Model (UBM)

## 4.1. Introduction

An Universal Background Model (UBM for short) is a large GMM trained to represent a set of speakers in a representative fashion of a given population. This is achieved by selecting a training data that is adequate to the application (for instance, gender-specific audio, or audio with specific capture type). For speaker diarization, the usual number of mixture models for a UBM is of 512/1024/2048. A large dataset is used with the Expectation Maximization algorithm until a convergence of some sort is achieved. The parameters for the mixture model are then stored [12].

## 4.2. Adaptation of Speaker Model

It is possible to derive a hypothesized speaker model by adapting the parameters of the UBM through a Maximum a Posteriori estimation. The previously trained UBM mixture model is used on the new observations on an Expectation step to generate new parameters for the mixture models. These new parameters are combined with the older parameters in a weighted fashion, considering the amount of new data and their probability as density estimators to cause “perturbations” on the previously converged mixtures. A single step of MAP is performed, and the resulting parameters of the models are stored as representatives of the speaker candidate. Usually, only the means of the model are concatenated to generate a new feature representation called a Supervector.

## 

## 5. Audio Analysis Library

## TODO

## 

## 6. Deep Learning for Audio Feature Extraction

### 

### 6.1. Introduction

Our initial investigations started with MFCCs and GMMs for Speaker Diarization. The majority of state-of-the-art implementations for audio tasks (e.g. speaker recognition, speaker diarization, speaker authentication) are based on GMMs and MFCCs. MFCCs are standard for any application related to audio, independently of the application. Based on such information, we proposed an investigation related to the application of Deep Learning, mainly Convolutional Neural Networks, for audio feature extraction. Our assumptions are the following:

* MFCCs and other hand-crafted features are too generic
* GMMs are being replaced by Deep Learning approaches in many scenarios
* Convolutional Neural Networks, mainly, are state-of-the-art for many applications
* CNNs are powerful feature extractors

Therefore, we believe that CNNs should yield good results for extracting features from audio. Our investigation started with selecting the most relevant papers related to deep learning applied for audio feature extraction. We started with the most recent state-of-the-art paper related to deep learning and audio: WaveNet. From such a paper, we selected three papers (plus one extra) to evaluate. Each of such papers will be discussed in the following sections. The last section has some experiments we performed with CNNs and audio.

### 

### 6.2. Papers Evaluated

### 

#### 6.2.1. WaveNet [6]

WaveNet is a recent implementation from Google that is the current state-of-the-art for generating raw audio waveforms. It is based on the application of dilated convolutions on raw audio and a lot of recent techniques. The generated audios are very similar to humans and it is very difficult to separate what is human and what is machine-generated. More information is available at: <https://deepmind.com/blog/wavenet-generative-model-raw-audio/>

One of the sections of this paper refers to deep learning being applied successfully to audio. Among such results, there are results related to audio feature extraction with deep learning, from where we selected four main papers. Such papers will be explored in the following sections.

#### 6.2.2. Estimating phoneme class conditional probabilities from raw speech signal using convolutional neural networks [7]

**Main Findings**

* Input is the raw audio. Outputs a phoneme class conditional probability;
* Uses a Convolutional Neural Network;
* Growing interests in using short-term spectrum as features;
* The raw features are composed of a windows of the speech signal. The window is normalized;
* Used TIMIT Corpus: 3,696 utterances from 462 speakers.

**Implementation**

The authors used a Convolutional Neural Network comprised of three convolutional layers and a fully-connected layers at the end, being the last one a softmax layer. The convolutional layers are composed of a convolution layer followed by a max pooling layer and the activation function is TANH.

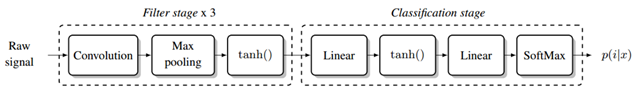


Figure shows the network architecture.

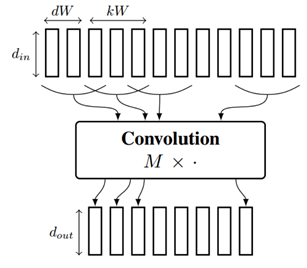


Figure shows details of the convolution over the raw audio signal. and are the dimension of the input and output frames. is the kernel width and is the shift between two linear applications.

The experimental setup created by the authors uses TIMIT Corpus. They tested many different variations on the experiments, using raw audio and MFCC with traditional MLPs as well as with CNNs. Some of the results are presented in the next figure.

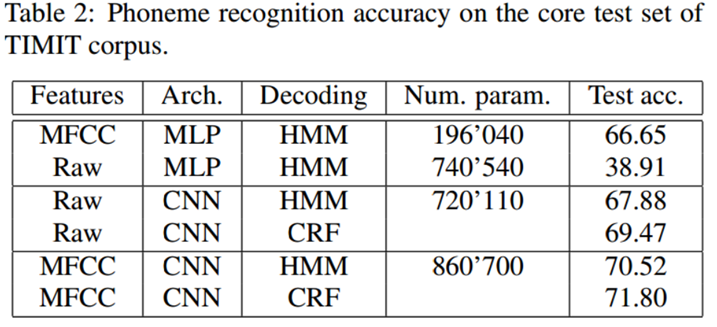


Figure shows results of phoneme recognition accuracy on the core test set of TIMIT corpus.

**Conclusion**

The results in the paper suggest that deep architectures are able to learn efficient features for estimating phoneme class probability. It was the first time that features learned by a deep learning architecture had similar performance to hand-crafted ones. The authors also argue that in a given moment, the advantages of using MFCC might collapse and deep architectures may be better for feature extraction.

#### 6.2.3. Acoustic Modeling with Deep Neural Networks Using Raw Time Signal for LVCSR [8]

**Main Findings**

* LVCSR - Large-Vocabulary Continuous Speech Recognition. Also known as Speech-to-Text, Full Transcription or Automatic Speech Recognition;
* Investigate DNN use for automatic speech recognition (ASR);
* Sigmoid to ReLU = decreased gap between MFCC and raw time signal;
* Used about 6 hidden layers with 2K neurons each;
* Input corresponds to 17 stacked frames;
* The output layers has 4500 neurons (phoneme classification);
* Trained on Quaero dataset (not found), comprised of 50 hours of speech in English;
* Used GMM/HMM as baseline.

**Implementation**

The authors used different hand-crafted features for comparison with Deep Neural Networks: Amplitude spectrum — FFT, Mel-Frequency cepstral coefficients — MFCC, Gammatone features — GT and Perceptual linear predictive coefficients — PLP. The Deep Neural Network developed contains six hidden layers of 2000 neurons each. The output layer contains around 4500 neurons, which corresponds to the generalized triphones tied by a phonetic classification and regression tree (CART). Some results and comparisons are in the next Figures.

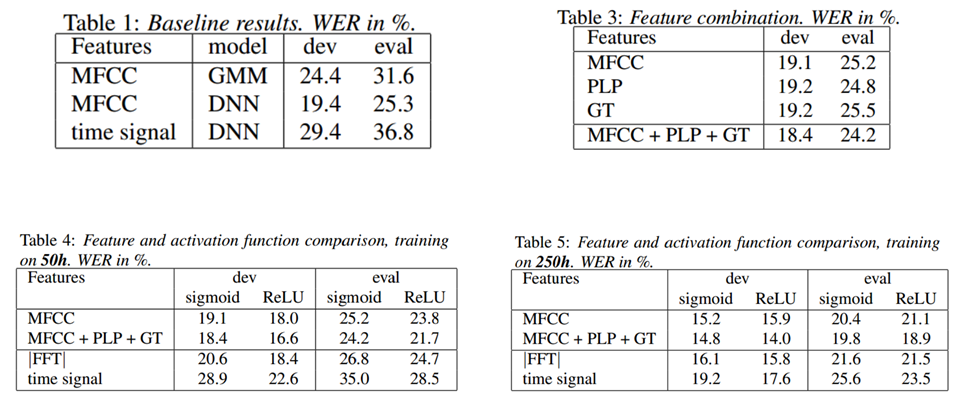


Figure shows tables with paper results. It is important to notice that in Table 4, ReLU instead of sigmoid yield good results (with 50h of training data), where Table 5 shows that, with around 250h of training data, the difference is reduced. Another important results is shown in Table 3, where the combination of MFCC, PLP and GT yielded a smaller WER when compared to each feature alone.

**Conclusion**

The results in the paper show that there is a reduction of the performance gap between using MFCC and raw audio when the activation of the network is ReLU. They also showed some interesting results with respect to the combination of hand-crafted features with Deep Neural Networks. In their experiments, such combination showed an improvement over them separately executed. It is also important to mention that a Deep Neural Network is able to learn a set of bandpass filters purely from raw signal.

### 

#### 6.2.4. Convolutional, Long Short-Term Memory, Fully Connected Deep Neural Networks [9]

**Main Findings**

* Combines CNNs, LSTMs and DNNs = CLDNN;
* Motivated by analysis that says that LSTM:
  + Can be improved with better input features (CNN);
  + Can be improved with deeper mapping between hidden and output layers (DNN).
* CNN layers: reduce spectral variation of the input;
* LSTM layers: reduce temporal variations;
* DNN layers: transform features into a space easier to classify.

**Implementation**

The authors combined convolutional layers with long short-term memory layers and fully-connected layers. Adding convolutional layers prior to LSTM ones yielded better results than LSTM layers by themselves. They also modeled the architecture to make use of short-term and long-term features. The dashed lines in the Figure below represent the implementation of such features, where is passed directly to the LSTM and to the fully-connected layers at the end.

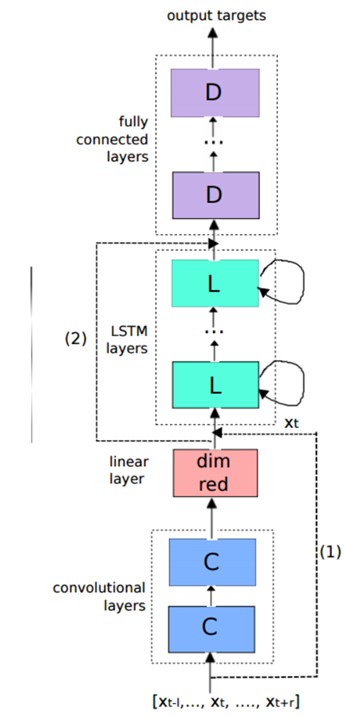


Figure shows the CLDNN architecture.

**Conclusion**

The main conclusion of this paper is that the proposed architecture (CLDNN) is able to produce a feature representation that is easily separable. It means that it is easier for a classifier to classify such representation instead of the ones used in related work. BEsides, the use of CNNs along with LSTMs yielded an improvement of 4-6% over only LSTMs.

### 

#### 6.2.5. Learning the Speech Front-end With Raw Waveform CLDNNs [10]

**Main Findings**

* They believe DNNs are more powerful than GMMs (feature extraction + classification);
* One of the difficulties in modeling the raw waveform is that perceptually and semantically identical sounds can appear at different phase shifts, so using a representation that is invariant to small phase shifts is critical;
* LSTM may not be suitable (25ms of data corresponds to 400 samples);
* Uses Convolutional, Long Short-Term Memory Deep Neural Network;
* First time raw waveform and MFCC match in performance!

**Implementation**

The authors used the same architecture of previous paper (CLDNN), but performed different experiments from previous paper. The input features for all baseline models are 40-dimensional log-mel filterbank features, computed every 10ms. All neural networks are trained with the cross-entropy criterion, using asynchronous stochastic gradient descent (ASGD) optimization

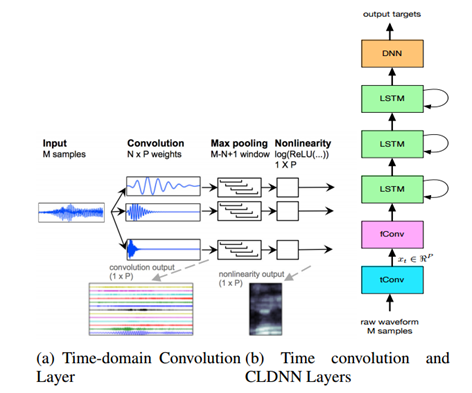


Figure shows the module of the raw waveform CLDNN.

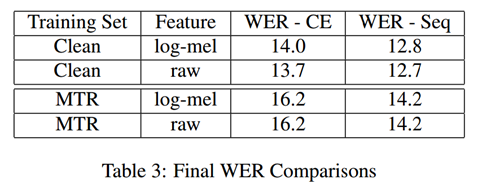


Figure shows some results with different training sets and different features and WER calculation. It is important to notice that MFCC and raw audio are very similar in result. Raw audio even bet the MFCC in Clean training set.

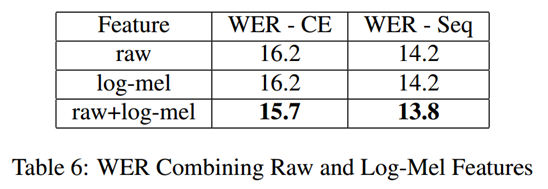


Figure shows the results when combining both raw audio and MFCC.

**Conclusion**

This paper has many interesting findings, but the most interesting one is related to raw waveform and MFCC match in performance for the first time. The implementation based on CLDNN yielded results with raw audio equal to those with MFCC. Besides, the authors also performed experiments with a combination of features extracted of the raw audio with features extracted from MFCC, which resulted in even better results.

### 

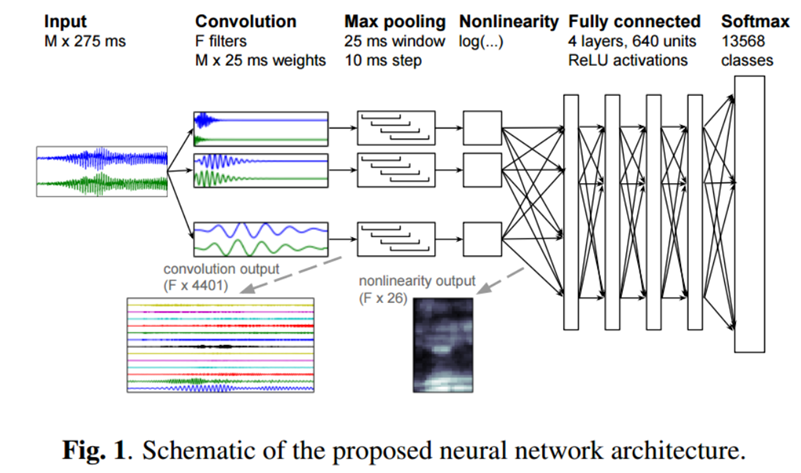
#### 6.2.6. Speech Acoustic Modeling From Raw Multichannel Waveforms [11]

**Main Findings**

* Uses a CNN-DNN approach to process multichannel audio;
* Outperforms log-mel filterbank magnitude features under noisy and reverberant conditions.

**Implementation**

The authors created a network that is able to “mimic” the steps involved in computing the MFCC features. With such an architecture, they are able to train an acoustic model that performs well without requiring hand-crafted features. Given a single-channel input, their architecture learns a representation qualitatively quite similar to MFCC features. The choice of operating in the time domain is further justified when using multichannel inputs which allows the network to learn spatial filters that filter out noise. Their architecture is able to learn jointly how to extract features and how to classify them.



**Conclusion**

The authors performed a variety of experiments related to multichannel waveforms. One of the main results they found is that, when noise comes from a consistent direction (a given region in the room), the network can improve recognition performance. When the noise is not from a specific region, the network’s accuracy drops. Additionally, the network is still slightly worse than the baselines, but there is a room for improvement, which is detailed in the paper conclusion.

#### 6.2.7. Conclusion

We noticed that many advances were made with deep learning applied to audio, mainly to feature extraction. Audio features are successfully extracted with deep learning methods, which is a great alternative to traditional hand-crafted features. It is important to mention that computational cost is something that must be weighted in the equation, mainly when we refer to real time applications. Another interesting insight is that LSTM by itself is not a good option, which means that a combination of convolutional layers along with LSTM layers is preferable.

The real challenge seems to be related to how to model the problem, choosing the right architecture, the window size and slicing as well as the hyperparameters of the network. However, we could notice that Speaker Diarization does not appear in any of these papers, so there are some opportunities for future research with Deep Learning applied to this specific domain.

## 

## Bibliography

[1] Mogran, Nelson, Hervé Bourlard, and Hynek Hermansky. "Automatic speech recognition: An auditory perspective." Speech processing in the auditory system. Springer New York, 2004. 309-338.

[2] Stevens, Stanley Smith; Volkmann; John and Newman, Edwin B. "A scale for the measurement of the psychological magnitude pitch". Journal of the Acoustical Society of America 8 (3), 1937 : 185–190.

[3] Harsha Yella, Sree and Stolcke, Andreas. "A Comparison of Neural Network Feature Transforms for Speaker Diarization", 2015.

[4] Anguera Miro, Xavier and Bozonnet, S. and Evans, Nicholas and Fredouille, Corinne and Friedland, G. and Vinyals, Oriol. "Speaker Diarization: A Review of Recent Research". IEEE Transactions on Audio, Speech, and Language Processing, 2012.

[5] Shum, Stephen H. and Dehak, Najim and Dehak, Reda and Glass, James R. "Unsupervised methods for speaker diarization: An integrated and iterative approach". IEEE Transactions on Audio, Speech, and Language Processing, 2013.

[6] van den Oord, Aäron, et al. "WaveNet: A generative model for raw audio." 9th ISCA Speech Synthesis Workshop, 2016.

[7] Palaz, Dimitri, Ronan Collobert, and Mathew Magimai Doss. "Estimating phoneme class conditional probabilities from raw speech signal using convolutional neural networks." arXiv preprint arXiv:1304.1018, 2013.

[8] Tüske, Zoltán, et al. "Acoustic modeling with deep neural networks using raw time signal for LVCSR." INTERSPEECH, 2014.

[9] Sainath, Tara N., et al. "Convolutional, long short-term memory, fully connected deep neural networks." 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2015.

[10] Sainath, Tara N., et al. "Learning the speech front-end with raw waveform CLDNNs." Proc. Interspeech, 2015.

[11] Hoshen, Yedid, Ron J. Weiss, and Kevin W. Wilson. "Speech acoustic modeling from raw multichannel waveforms." 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2015.

[12] Douglas A. Reynolds, “Speaker Verification Using Adapted Gaussian Mixture Models”, Digital Signal Processing, 2010