UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE
AGH

DEEP NEURAL NETWORKS IN ACOUSTIC MODEL

A Degree Thesis

Submitted to the Faculty of the

Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

and

AGH University of Science and Technology

by

ORIOL CAMACHO TEJEDOR

In partial fulfilment

of the requirements for the degree in

AUDIOVISUAL SYSTEMS ON TELECOMMUNICATIONS ENGINEERING

Advisor: Bartosz Ziółko

Advisor: José Adrián Rodríguez Fonollosa

May 2016

Abstract

Nowadays the systems of speech recognition are many communes between mobiles and computers. Increasingly they are more efficient, useful and more common between users, so we can say the ASR is a very comfortable way to communicate with hardware of every machine. There exist several methods to the speech recognition, in this thesis is studied the DNN.

DNN is a system of machine learning based in hidden states. For the implementation we use kaldi library. The first part of this thesis, is a theoretical study of machine learning and the DNN method. The second part is about the implementation and improvement of DNN system.

Resum

Els sistemes de reconeixement de la parla, actualment son molt comuns als mòbils, pc,... Cada cop son mes eficients, útils y utilitzes pels usuaris. Podríem dir que els ASR es una forma molt còmoda de comunicar-nos amb el hadware, de qualsevol maquina. Hi han varis mètodes de ASR, en aquest treball s'estudia el DNN, una tècnica molt moderna que treballa amb estats ocults, i bases de dades. Per a implementar-lo., utilitzarem la llibreria Kaldi. La primera part del treball, es fa un estudi de machine learning y el mètode DNN, de una forma teòrica. En la segona part, es la implementació I millora de un sistema DNN.

Resumen

Los sistemas de reconocimiento del habla, actualmente son muy comunes en los móviles, pc,... Cada vez son mas eficientes, útiles y mas utilizados por los usuarios. Podríamos decir que los ASR es una forma muy cómoda de comunicarnos con el hardware, de cualquier maquina. Hay varios métodos para el reconocimiento del habla, en este trabajo se estudia el DNN, es un sistema de machine learning basado en estados ocultos. Para implementarlo se utiliza la libraría kaldi. La primera parte del trabajo, se hace un estudio de machine learning y el método DNN, de una forma teórica, En la segunda parte, es la implementación y mejora de un sistema DNN.

Acknowledgements

A special thanks to Prof. Bartosz Bziolko who has give the opportunity to work in his laboratory these months in this project. I am also grateful to Prof. Jose Adrian Rodiguez Fonollosa, for his assistance and supervision. Finally I thank to my family for their continuous support, during all this months.

Revision history and approval record

Revision	Date	Purpose
0	1/04/2016	Document creation
1	13/05/2016	Document revision

DOCUMENT DISTRIBUTION LIST

Name	e-mail
Oriol Camacho	ocamacho198@gmail.com
Bartosz Bziolko	bziolko@agh.edu.pl
José Adrián Rodríguez Fonollosa	jose.fonollosa@upc.edu

Written by: Oriol camacho		Reviewed and approved by:	
Date	16/05/2016	Date	16/05/2016
Name	Oriol Camacho Tejedor	Name	Bartosz Bziolko
Position	Project Author	Position	Project Supervisor

Table of contents

List of Figures

List of Tables:

1. Introduction

Automatic Speech Recognition (ASR) is a disciplinary of the machine learning, also called artificial intelligence, which has as main goal allow the oral communication between humans and computers.

1.1. Formulation of the problem

The main problem in the ASR is the complexity of the human language. This is due to are have a lot of languages and within each language are many forms of dialects, accents, types of pronunciations, so on. All this things influence the complexity, that it must have the ASR systems.

Nowadays the ASR is one of the most often used for the users of mobiles, tablets, so on. This all open a opportunity and a possibility in the market, the major companies of technological like Google, Apple, Samsung, so on, expend a lot of resources in this topic.

There are a lot of kind of ASR system and methods, in this thesis is used the Deep Neural Networks (DNN), one of the most actuality machine learning method, based in Neural Networks.

In our ASR system, we use Kaldi, a toolkit for speech recognition written in C++ and licensed under the Apache License v2.0.

1.2. Statement of purpose

The purpose of this project is give a system of ASR based in Deep Neural Networks (DNN), learn and improve about both ASR and Kaldi toolkit, to be able to build an Automatic Speech Recognition system.

The project main goals are:

1. Do a theoretical study these following topics: ASR DNN, and Kaldi toolkit
2. Training a data base for
3. Develop a system for DNNs
4. Testing
5. Improve
6. Pretend to create a Graphical User Interface to test our system

1.3. Project background

The project is carried out at the Signal processing group in AGH University of Science and Technology. The supervisor Bartosz Ziólko provides the main project initial ideas. Although the development of it is carried out by both the supervisor and I together.

1.4. Project outline

This project is divided into three main sections, the first serious explain the basics, which I have drawn to the project, that is the basics of ASR, the Kaldi library, and an introduction to machine learning. The second section, the project was explained in general, Deep Neural Networks (DNN) and its development. In the third section, we see the results, and better analysis of this project. Lastly, remember that after the results have sections of the conclusions, and budget.

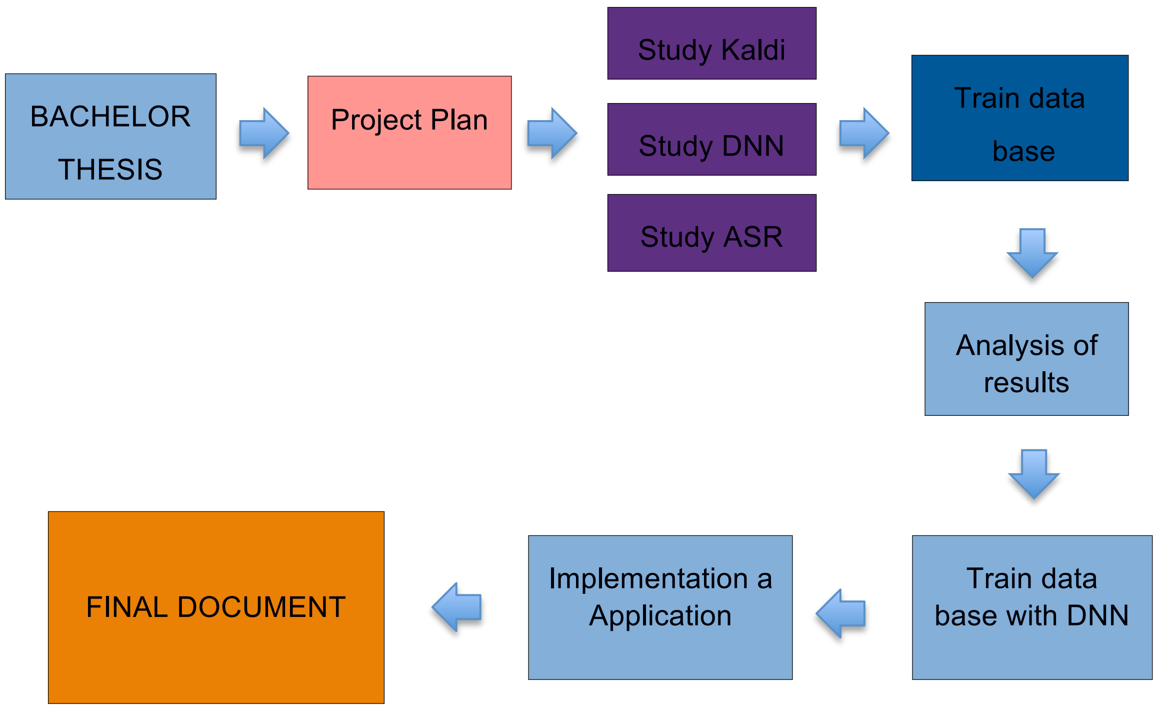
1.5. Requirements and specifications

The requirements of this project are:

- Advanced Knowledge in Audio processing
- Advanced Knowledge in Mathematics
- Knowledge in machine learning
- Experience with C++
- Individual Skills
- Good level of English
- Experience with ubuntu
- Experience with python
- Experience with shell/bash
- C++ Libraries
- Audio Database recorded

The project includes these C++ libraries: OpenCV, Lapack, Eigen and Boost. The project is organized and structured in different sections; each part develops a different goal.

1.6 Project plan



2. Basics concepts of ASR State of the art of the technology used in this thesis:

A brief review of the behaviour and state of the art of the ASR and Signal analysis is presented. This is followed by a presentation of Kaldi Toolkit and it ends with a summary the different kinds of machine learning.

2.1. Automatic Speech Recognition

The automatic speech recognition (ASR) is simply the translation of spoken words into text. In this section, the key components of an ASR system are broadly presented [1].

Acoustic Features: A raw audio signal, for example as received from a microphone, needs to be converted into a more manageable. First the incoming audio is treated as a sequence of frames at regular time intervals. These frames are then analysed such that some data can be extracted. For each frame, we obtain a representative feature vector [1].

Language Model (LM): A language model describes how words can be combined

Acoustic Model (AM): An acoustic model contains the data describing the acoustic nature of all the phonemes understood by the system. Acoustic models are built through a training process using large quantities of transcribed audio. Each context dependent phoneme, also called triphone, is represented by a hidden Markov model (HMM). The HMM states permit to describe how the sound of the phonemes progresses in time [1].

Decoder: The decoder is the reason of the ASR system; its job is to decode a sequence of speech signal to reveal what words were spoken. For each audio frame, there is a process of pattern matching [1].

The statistical approach to automatic speech recognition aims at modelling the stochastic relation between a speech signal and the spoken word sequence with the objective of

Minimizing the expected error rate of a classifier. The statistical paradigm is governed by Bayesian decision.

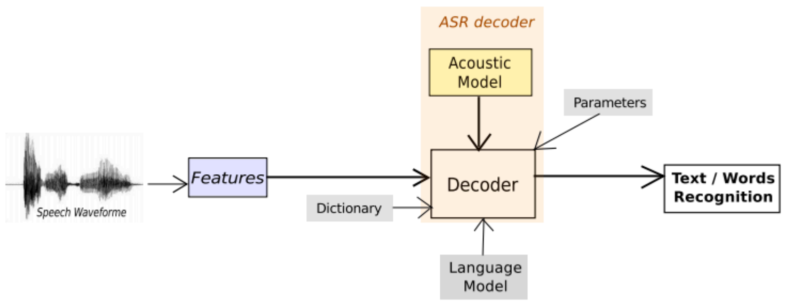


Figure 1: Overview of the ASR system [1]

Bayesian decision theory is a fundamental statistical approach to the problem of pattern classification [2]. This approach is based on quantifying the trade offs between various classification decisions using probability and the costs that accompany such decisions. It makes the assumption that the decision problem is posed in probabilistic terms, and that all of the relevant probability values are known.

Bayes' theorem is stated mathematically as the following equation [2]:

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)},$$

where *A* and *B* are events

- *P*(*A*) and *P*(*B*) are the probabilities of *A* and *B* without regard to each other.
- *P*(*A*|*B*), a conditional probability, is the probability of observing event *A* given that *B* is true.
- *P*(*B*|*A*) is the probability of observing event *B* given that *A* is true.

The statistical paradigm is governed by Bayes' decision rule: Given a sequence of acoustic observations $x_1^T = x_1, \ldots, x_T$ as the constituent features of a spoken utterance, Bayes' decision rule decided for that word $w_1^N = w_1, \ldots, w_N$ sequence which maximizes the class posterior probability $p(w_1^N|x_1^T)$ [3]:

$$[w_1^N]_{\text{opt}} = \underset{w_1^N}{\text{argmax}} \{p(w_1^N|x_1^T)\}.$$

In automatic speech recognition, the generative model, which decomposes the class posterior probability into a product of two independent stochastic knowledge sources,

became widely accepted[3]:

$$p(w_1^N|x_1^T) = \frac{p(w_1^N) \cdot p(x_1^T|w_1^N)}{p(x_1^T)}.$$

the decision rule is equivalent to[3]:

$$[w_1^N]_{\text{opt}} = \underset{w_1^N}{\text{argmax}} \{p(w_1^N) \cdot p(x_1^T|w_1^N)\}.$$

The word sequence, which maximizes the posterior probability, is determined by searching for that word sequence which maximizes the product of the following two stochastic knowledge sources:

• The acoustic model $p(w_1^N|x_1^T)$ which captures the probability of observing a sequence of acoustic x_1^T observations given a word sequence w_1^N [3].

• The language model $p(w_1^N)$ which provides a prior probability for the word sequence w_1^N [3].

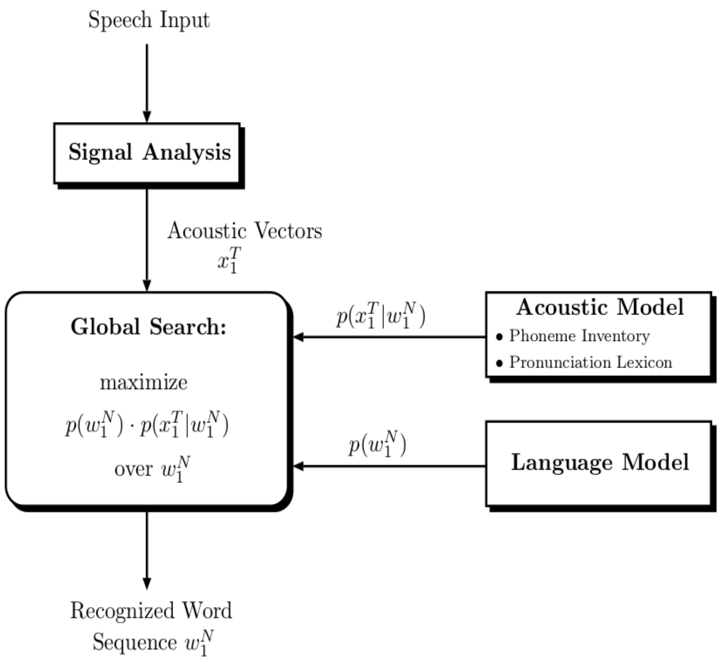


Figure 2: Basic architecture of statical ASR system [3]

2.2. Signal Processing

The first step in any automatic speech recognition system is to extract features. A speech signal processing does features extraction. Its aim is to provide a compact encoding of the speech waveform. This encoding should minimize the loss information and provide a good match with the distributional assumptions made by the acoustic models. The final result is a feature vector in Rn. Many features extraction techniques are available, these include [1]:

•Linear predictive cepstral coefficients (LPCC)

•Mel-frequency cepstral coefficients (MFCCs)

•Perceptual linear predictive coefficients (PLP)

Our feature extraction and waveform-reading code aims to create standard MFCC and PLP features, setting reasonable defaults but leaving available the options that people are most likely to want to tweak [4].

The overall MFCC computation is as follows [4]:

- Work out the number of frames in the file (typically 25 ms frames shifted by 10ms each time).
- For each frame:
    - Extract the data, do optional dithering, preemphasis and dc offset removal, and multiply it by a windowing function (various options are supported here, e.g. Hamming)
    - Work out the energy at this point
    - Do FFT and compute the power spectrum
    - Compute the log of the energies and take the cosine transform, keeping as many coefficients as specified
    - Optionally do cepstral liftering; this is just a scaling of the coefficients, which ensures they have a reasonable range.

The lower and upper cut-off of the frequency range covered by the triangular mel bins are controlled by the options –low-freq and –high-freq, which are usually set close to zero and the Nyquist frequency respectively, e.g. –low-freq=20 and –high-freq=7800 for 16kHz sampled speech [4].

Transforms, projections and other feature operations that are typically not speaker specific include:

• Frame splicing and Delta feature computation[5].

• Linear Discriminant Analysis (LDA) transform[6].

• Heteroscedastic Linear Discriminant Analysis (HLDA).

• Maximum Likelihood Linear Transform (MLLT) estimation[7].

Delta feature computation

MFCC feature only takes account of the relationship in phonetic frames without considering the relationship between them. Phonetic signals are essentially continuous, so the acquisition of the dynamic changing feature between phonetic frames will improve the performance of recognition.

LDA+MLLT

LDA: Is a linear transform that reduce dimensionality of our input features. The idea of LDA is to find a linear transformation of feature vectors from an n-dimensional space to vectors in an m-dimensional space (m<n) such that the class separability is maximum.

MLLT: Estimates the parameters of a linear transform in order to maximize the likelihood of the training data given a diagonal-covariance Gaussian mixture models; the transformed features are better represented by the model than the original features.

2.3. Acoustic Model (AM)

The acoustic model $p(w_1^N|x_1^T)$ provides a stochastic description for the realization of a sequence of acoustic observation vectors x_1^T given a word sequence w_1^N. Due to data sparsely, the model for individual words as well as the model for entire sentences is obtained by concatenating the acoustic models of basic sub-word units according to a pronunciation lexicon. Word units smaller than words enable a speech recognizer to allow for recognizing words that do not occur in the training data. Thus, the recognition system can ensure that enough instances of each sub-word unit have been observed in training to allow for a reliable estimation of the underlying model parameters [3]

The temporal distortion of different pronunciations as well as the spectral variation in the acoustic signal can be described via a Hidden Markov Model (HMM). A HMM is a stochastic finite state automaton that models the variation in the acoustic signal via a two-stage stochastic process. The automaton is defined through a set of states with transitions connecting the states. The probability $p(w_1^N|x_1^T)$ is extended by an unobservable variable s representing the states[3]:

$$p(x_1|w_1^N) = \sum_{s_1^T} p(x_1^T, s_1^T|w_1^N).$$

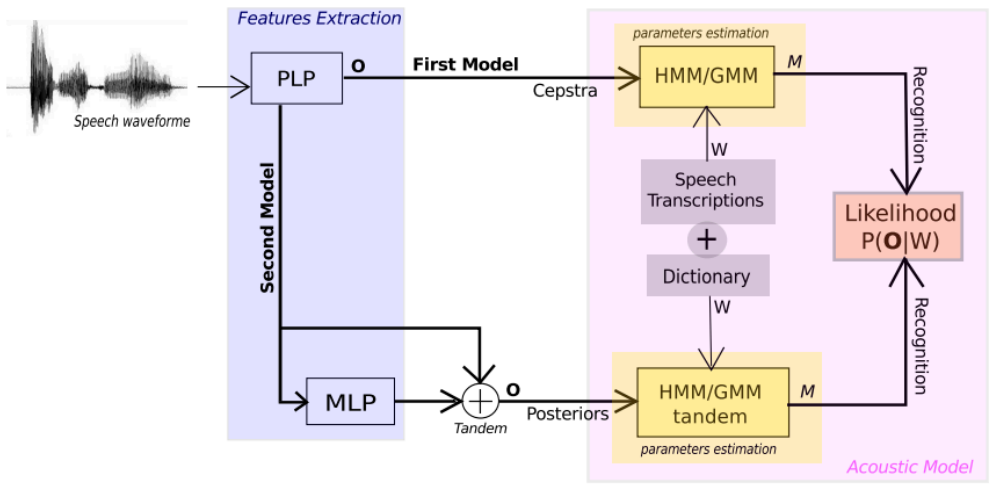


Figure 3:Architecture of the acoustic model (AM) [1]

Gaussian mixture models

We support GMMs with diagonal and full covariance structures. Rather than representing individual Gaussian densities separately, we directly implement a GMM class that is parameterized by the natural parameters, i.e. means times inverse covariances and inverse covariances [8].

GMM-based acoustic model

The “acoustic model” class AmDiagGmm represents a collection of DiagGmm objects, indexed by “pdf-ids” that correspond to context-dependent HMM states. This class does not represent any HMM structure, but just a collection of densities (i.e. GMMs).[8]

Speaker adaptation

We support both model-space adaptation using maximum likelihood linear regression (MLLR) [8] and feature-space adaptation using feature-space MLLR (fMLLR), also known as constrained MLLR [9]. For both MLLR and fMLLR, multiple transforms can be estimated using a regression tree [10]. When a single fMLLR transform is needed, it can be used as an additional processing step in the feature pipeline. The toolkit also supports speaker normalization using a linear approximation to VTLN, similar to [11], or conventional feature-level VTLN, or a more generic approach for gender normalization, which we call the “exponential transform” [12]. Both fMLLR and VTLN can be used for speaker adaptive training (SAT) of the acoustic models.

2.4. Language Model (LM)

Language models are used to constrain search in a decoder by limiting the number of possible words that need to be considered at any one point in the search. The consequence is faster execution and higher accuracy.

Language models constrain search either absolutely (by enumerating some small subset of possible expansions) or probabilistically (by computing a likelihood for each possible successor word). The former will usually have an associated grammar this is compiled down into a graph, the latter will be trained from a corpus.

Statistical language models (SLMs) are good for free-form input, such as dictation or spontaneous speech, where it's not practical or possible to a priori specify all possible legal word sequences.

Trigram SLMs are probably the most common ones used in ASR and represent a good balance between complexity and robust estimation. A trigram model encodes the probability of a word (w3) given its immediate two-word history, ie p(w3 | w1 w2). In practice trigam models can be "backed-off" to bigram and unigram models, allowing the decoder to emit any possible word sequence (provided that the acoustic and lexical evidence is there)[9].

2.5 Kaldi

As is mentioned above, Kaldi is an open-source toolkit for speech recognition written in C++ and licensed under the Apache License v2.0. Kaldi is intended for use by speech recognition researchers. The principal goal of Kaldi is to have modern and flexible code that is easy to understand, modify and extend[4].

In Kaldi open-source toolkit exist several potential choices for building a recognition system. Important feactures include:

- Code-level integration with Finite State Transducers (FSTs)

- Extensive linear algebra support

- Extensible design

- Thorough testing

- Open license: The code is licensed under Apache 2.0, which is one of the least restrictive licenses available

- Complete recipes

The toolkit depends on two external libraries: one is OpenFst for the finite-state framework, and the other is a numerical algebra library. The Kaldi use the Basic Linear Algebra Subroutines (BLAS) and Linear Algebra PACKage (LAPACK).

The feature extraction and waveform-reading code aims to create MFCC and PLP features. Kaldi used feature extraction approaches: e.g. VTLN, cepstral mean and variance normalization, LDA, STC/MLLT, HLDA, and so on [8].

Kaldi to support conventional models, diagonal GMMs and Subspace Gaussian Mixture Models (SGMMs).

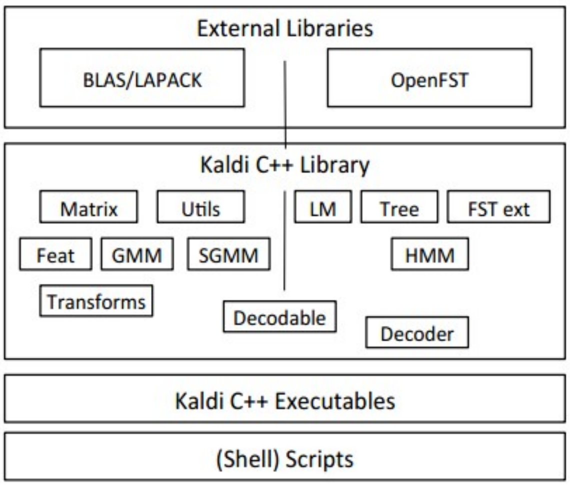


Figure 4: Kaldi [8]

A simplified view of the different components of Kaldi. The library modules can be grouped into those that depend on linear algebra libraries and those that depend on OpenFst. The decodable class bridges these two halves. Modules that are lower down in the schematic depend on one or more modules that are higher up[8]

2.6 INTRO MACHINE LEARNING (DNN)

Machine learning is a subfield of computer science that evolved from the study of pattern recognition and computational learning theory in artificial intelligence. Machine learning explores the study and construction of algorithms that can learn from and make predictions on data. Such algorithms operate by bulding a model from example inputs in order to make data-driven predictions or decisions expressed as outputs[11].

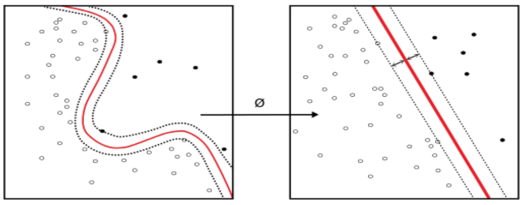


Figure 5: Machine learning [11]

Deep learning (deep structured learning, hierarchical learning or deep machine learning) is a branch of machine learning based on a set of algorithms that attempt to model high-level abstractions in data by using multiple processing layers, with complex structures or otherwise, composed of multiple non-linear transformations.

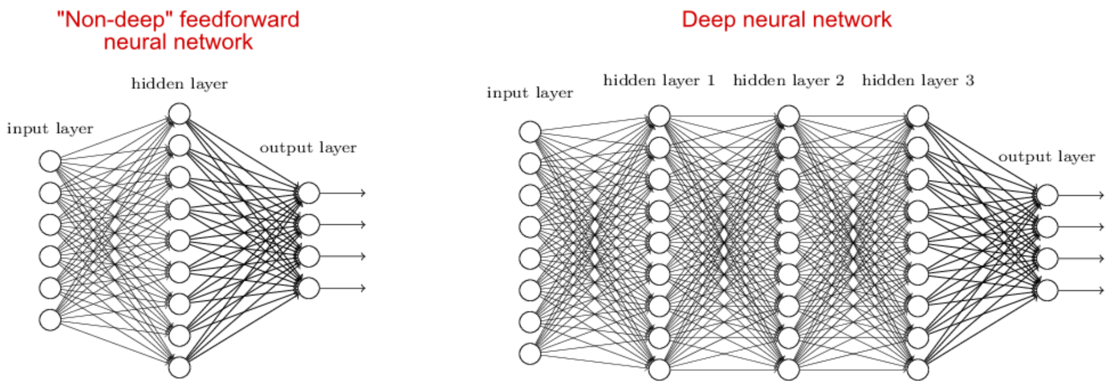


Figure 6: DNN [11]

3. Deep Neural Networks:

3.1 Introduction:

New machine learning algorithms can lead to significant advances in automatic speech recognition (ASR). The biggest single advance occurred nearly four decades ago with the introduction of the expectation-maximization (EM) algorithm for training HMMs[11][12]. With the EM algorithm, it became possible to develop speech recognition systems for real world tasks using the richness of GMMs to represent the relationship between HMM states and the acoustic input. In these systems the acoustic input is typically represented by concatenating Mel-frequency cepstral coefficients (MFCCs) or perceptual linear predictive coefficients (PLPs)[14]. This nonadaptive but highly engineered preprocessing of the waveform is designed to discard the large amount of information in waveforms that is considered to be irrelevant for discrimination and to express the remaining information in a form that facilitates discrimination with GMM-HMMs[10].

GMMs have a serious shortcoming they are statistically inefficient for modeling data that lie on or near a nonlinear manifold in the data space. For example, modeling the set of points that lie very close to the surface of a sphere only requires a few parameters using an appropriate model class, but it requires a very large number of diagonal Gaussians or a fairly large number of full-covariance Gaussians[10]. Speech is produced by modulating a relatively small number of parameters of a dynamical system, and this implies that its true underlying structure is much lower-dimensional than is immediately apparent in a window that contains hundreds of coefficients.

Therefore, that other types of model may work better than GMMs. Artificial neural networks trained by backpropagating error derivatives have the potential to learn much better models of data that lie on or near a nonlinear manifold[10].

Over the last few years, advances in both machine learning algorithms and computer hardware have led to more efficient methods for training DNNs [10] that contain many layers of nonlinear hidden units and a very large output layer.

Using the new learning methods, several different research groups have shown that DNNs can outperform GMMs at acoustic modeling for speech recognition on a variety of data sets including large data sets with large vocabularies[10].

This section starts by describing the two-stage training procedure that is used for fitting the DNNs. In the first stage, layers of feature detectors are initialized, one layer at a time, by fitting a stack of generative models, each of which has one layer of latent variables.

These generative models are trained without using any information about the HMM states that the acoustic model will need to discriminate. In the second stage, each generative model in the stack is used to initialize one layer of hidden units in a DNN and the whole network is then discriminatively fine-tuned to predict the target HMM states[10].
These targets are obtained by using a baseline GMM-HMM system to produce a forced alignment. In this article, we review exploratory experiments on the voxforge database, that were used to demonstrate the power of this two-stage training procedure for acoustic modeling.

3.2 TRAINING DEEP NEURAL NETWORKS

A deep neural network (DNN) is a feed-forward, artificial neural network that has more than one layer of hidden units between its inputs and its outputs. Each hidden unit, j, typically uses the logistic function to map its total input from the layer below, x_j, to the scalar state, y_j that it sends to the layer above[10].

$$y_j = logistic(x_j) = \frac{1}{1+e^{-x_j}}, \qquad x_j = b_j + \sum_i y_i w_{ij},$$

Where b_j is the bias of unit j, i is an index over units in the layer below, and w_{ij} is a the weight on a connection to unit j from unit i in the layer below. For multiclass classification, output unit j converts its total input, x_j , into a class probability, p_j, by using the "softmax" non-linearity[10] :

$$p_j = \frac{\exp(x_j)}{\sum_k \exp(x_k)},$$

where k is an index over all classes.

For large training sets, it is more efficient to compute the derivatives, random "mini-batch", rather than the whole training set, before updating the weights in proportion to the gradient. This stochastic gradient descent method can be further improved by using a "momentum" coefficient, 0< α <1, that smooths the gradient computed for mini-batch t [10].

$$\Delta w_{ij}(t) = \alpha \Delta w_{ij}(t-1) - \epsilon \frac{\partial C}{\partial w_{ij}(t)}.$$

To reduce overfitting, large weights can be penalized in proportion to their squared magnitude, or the learning can simply be terminated at the point at which performance on a held-out validation set starts getting worse[14]. In DNN, the initial weights are given small random values to prevent all of the hidden units in a layer from getting the same gradient

DNN with many hidden layers are hard to optimize.

This makes them capable of modeling very complex and highly non-linear relationships between inputs and outputs. This ability is important for high-quality acoustic modeling, but it also allows them to model spurious regularities that are an accidental property of the particular examples in the training set, which can lead to severe overfitting. Weight penalties or early-stopping can reduce the overfitting but only by removing much of the modeling power. Very large training sets [15] can reduce overfitting whilst preserving modeling power, but only by making training very computationally expensive. What we need is a better method of using the information in the training set to build multiple layers of non-linear feature detectors.

Generative pre-training

Instead of designing feature detectors to be good for discriminating between classes, can start by designing them to be good at modeling the structure in the input data. The idea is to learn one layer of feature detectors at a time with the states of the feature detectors in one layer acting as the data for training the next layer. After this generative “pre-training”, the multiple layers of feature detectors can be used as a much better starting point for a discriminative “fine-tuning” phase during which backpropagation through the DNN slightly adjusts the weights found in pre-training [16]. Some of the high-level features created by the generative pre-training will be of little use for discrimination, but others will be far more useful than the raw inputs. The generative pre-training finds a region of the weight-space that allows the discriminative fine-tuning to make rapid progress, and it also significantly reduces overfitting [17].

There are two classes of generative model. A directed model generates data by first choosing the states of the latent variables from a prior distribution and then choosing the states of the observable variables from their conditional distributions given the latent states[10].

An undirected model has a very different way of generating data. Instead of using one set of parameters to define a prior distribution over the latent variables and a separate set of parameters to define the conditional distributions of the observable variables given the values of the latent variables, an undirected model uses a single set of parameters, W, to define the joint probability of a vector of values of the observable variables, v, and a vector of values of the latent variables, h, via an energy function,

E:

$$p(\mathbf{v},\mathbf{h};\mathbf{W}) = \frac{1}{Z}e^{-E(\mathbf{v},\mathbf{h};\mathbf{W})}, \quad Z = \sum_{\mathbf{v}',\mathbf{h}'} e^{-E(\mathbf{v}',\mathbf{h}';\mathbf{W})},$$

where Z is called the “partition function”.

If many different latent variables interact non-linearly to generate each data vector, it is difficult to infer the states of the latent variables from the observed data in a directed model because of a phenomenon known as “explaining away” [18]. In undirected models, however, inference is easy provided the latent variables do not have edges linking them. Such a restricted class of undirected models is ideal for layerwise pre-training because each layer will have an easy inference procedure.

We start by describing an approximate learning algorithm for a restricted Boltzmann machine (RBM) which consists of a layer of stochastic binary “visible” units that represent binary input data connected to a layer of stochastic binary hidden units that learn to model significant non-independencies between the visible units [19].

An efficient learning procedure for RBMs

A joint configuration, (v,h) of the visible and hidden units of an RBM has an energy given by[10]:

$$E(\mathbf{v},\mathbf{h}) = -\sum_{i\in\text{visible}} a_i v_i - \sum_{j\in\text{hidden}} b_j h_j - \sum_{i,j} v_i h_j w_{ij}$$

where v_i, h_j are the binary states of visible unit i and hidden unit j, a_i , b_j are their biases and w_{ij} is the weight between them.

The network assigns a probability to every possible pair of a visible and a hidden vector via this energy functio, is given by summing over all possible hidden vectors [10] :

$$p(\mathbf{v}) = \frac{1}{Z}\sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}$$

The derivative of the log probability of a training set with respect to a weight is surprisingly simple[10]:

$$\frac{1}{N}\sum_{n=1}^{n=N}\frac{\partial \log p(\mathbf{v}^n)}{\partial w_{ij}} = <v_i h_j>_{data} - <v_i h_j>_{model}$$

The simple derivative in following equation leads to a very simple learning rule for performing stochastic steepest ascent in the log probability of the training data

$$\Delta w_{ij} = \epsilon(<v_i h_j>_{data} - <v_i h_j>_{model})$$

where ϵ is a learning rate

Getting an unbiased sample of $<v_i h_j>_{model}$, however, is much more difficult. It can be done by starting at any random state of the visible units and performing alternating Gibbs sampling for a very long time.

A much faster learning procedure called “contrastive divergence” (CD) was proposed in [19]. Finally, the states of the hidden units are updated again. The change in a weight is then given by [10]

A simplified version of the same learning rule that uses the states of individual units instead of pairwise products is used for the biases.

Contrastive divergence works well even though it is only crudely approximating the gradient of the log probability of the training data [19].

RBM learn better generative models if more steps of alternating Gibbs sampling are used before collecting the statistics for the second term in the learning rule, but for the purposes of pre-training feature detectors.

$$\Delta w_{ij} = \epsilon(<v_i h_j>_{data} - <v_i h_j>_{recon})$$

To suppress noise in the learning, the real-valued probabilities rather than binary samples are generally used for the reconstructions and the subsequent states of the hidden units, but it is important to use sampled binary values for the first computation of the hidden states because the sampling noise acts as a very effective regularizer that prevents overfitting [20].

Modeling real-valued data

Real-valued data, such as MFCCs, are more naturally modeled by linear variables with Gaussian noise and the RBM energy function can be modified to accommodate such variables, giving a Gaussian-Bernoulli RBM (GRBM)[10]:

$$E(\mathbf{v},\mathbf{h}) = \sum_{i\in\text{vis}} \frac{(v_i - a_i)^2}{2\sigma_i^2} - \sum_{j\in\text{hid}} b_j h_j - \sum_{i,j} \frac{v_i}{\sigma_i} h_j w_{ij}$$

where σ_i is the standard deviation of the Gaussian noise for visible unit i. The two conditional distributions required for CD learning are[10]:

$$p(h_j|\mathbf{v}) = logistic\left(b_j + \sum_i \frac{v_i}{\sigma_i} w_{ij}\right)$$

$$p(v_i|\mathbf{h}) = \mathcal{N}\left(a_i + \sigma_i \sum_j h_j w_{ij},\quad \sigma_i^2\right)$$

where $\mathcal{N}(\mu, \sigma^2)$ is a Gaussian. Learning the standard deviations of a GRBM is problematic for reasons described in [20],

Stacking RBMs to make a deep belief network

After training an RBM on the data, the inferred states of the hidden units can be used as data for training another RBM that learns to model the significant dependencies between the hidden units of the first RBM. The RBM in a stack can be combined in a surprising way to produce a single, multi-layer generative model called a deep belief net (DBN) [21].

To understand how RBMs are composed into a DBN it is helpful to rewrite this eq. the and to make explicit the dependence on W

$$p(\mathbf{v};\mathbf{W}) = \sum_{\mathbf{h}} p(\mathbf{h};\mathbf{W})p(\mathbf{v}|\mathbf{h};\mathbf{W}),$$

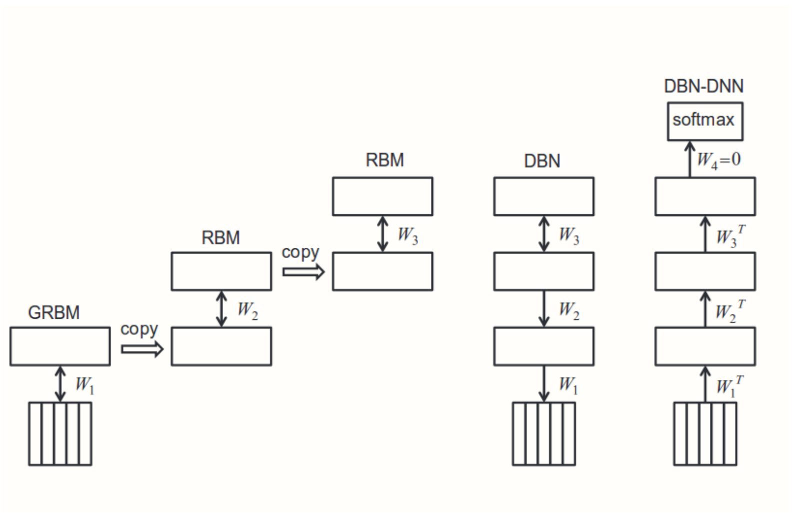


Figure 7: RBM [10]

There is a series of variational bounds on the log probability of the training data, and furthermore, each time a new RBM RBM is added to the stack, the variational bound on the new and deeper DBN is better than the previous variational bound [21], provided the new RBM is initialized and learned in the right way. While the existence of a bound that keeps improving is mathematically reassuring, it does not answer the practical issue, addressed in this review paper, of whether the learned feature detectors are useful for discrimination on a task that is unknown while training the DBN.

3.3 Acoustic Model training

Estimation of HMM parameters is commonly performed according to the Maximum Likelihood Estimation (MLE) criterion, which maximizes the probability of the training samples with regard to the model. This is done by applying the Expectation-Maximization (EM) algorithm, which relies on maximizing the log-likelihood from incomplete data, by iteratively maximizing the expectation of log-likelihood from complete data[26].

3.4 Acoustic data

All data used in our training experiments comes from VoxForge project [9]. It was setup to collect transcribed speech for use with Free and Open Source Speech Recognition Engines. They make available all submitted audio files that VoxForge user record under the GPL license. Our experiments will train and test our acoustic models just with English data [9].

In the following the dataset used:

Data Base	Speakers	Files	Time [minutes]	second/speakers
Train	358	15264	1272	213.18
Test	20	399	33.25	99.75

Table 1: Data base VoxForge

Each file or sentence lasts 5 seconds, therefore the database are intended to train 1,272 minutes recording

4. Experimental testing and Results

In this section we are going to show all the results and analysis for a different acoustic methods of training, which we presented in the previous sections. First sable, we will see the results of database voxforge training, then we can see the experiments with training DNN, and after then will be see the app presented in previous sections.

4.1 Evaluation

Have different methods to evaluate the quality of an ASR system. Word Error Rate (WER) is the most common metric of the performance of speech recognition. Therefore, we used WER for all our experiments and results.

The general difficulty of measuring performance lies in the fact that the recognized word sequence can have a different length from the reference word sequence (supposedly the correct one). The WER is a valuable tool for comparing different systems as well as for evaluating improvements within one system.

$$WER = \frac{S + D + I}{N}$$

where

- *S* is the number of substitutions,
- *D* is the number of deletions,
- *I* is the number of insertions,
- *N* is the number of words in the reference (N=S+D+C)

Alignment example:

REF:	portable	****	phone	upstairs	last	night	so	***
HYP:	preferable	form	of	stores	next	light	so	far
Eval	S	I	S	S	S	S		I

WER = 100 (1+5+1)/6 = 117%

4.2 Acoustic modelling scripts

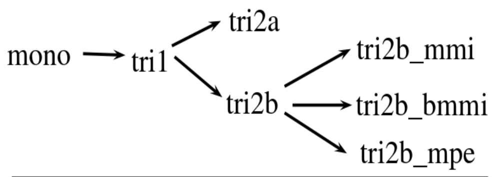
The recordings and their transcriptions from training dataset are used for acoustic modelling. The estimated AMs are evaluated on the test set. The decoding of the test utterances is performed always with the same parameters, so that different AMs can be compared. The used methods are listed in Figure 8 together with their hierarchy. The hierarchy shows that a more advanced method typically reuses initial values from previously trained simpler AM. At first, a mono-phone model is trained from flat start using the MFCCs, Δ and $\Delta\Delta$ features. We force-align the feature vectors to HMM states using utterances transcriptions. Secondly, we retrain the triphone AM (tri1a)[25].

One branch of experiments finishes by training MFCC Δ+$\Delta\Delta$ triphone AM (tri2a). On the other hand, the second branch instead of $\Delta + \Delta\Delta$ transformation uses LDA+MLLT to train AM (tri2b). Using the AM tri2b three AMs are discriminatively trained using the following objective functions[25]:

1. Maximum Mutual Information[22]. The model tri2b_mmi is trained in four loops.

2. Boosted Maximum Mutual Information[24]. The model tri2b_bmmi is trained in four loops with parameter 0.05.

3. Minimum Phone Error[23]. The model tri2b_mpe is also retrained in four loops



Training method name	Script shortcut
Monophone	mono
Triphone	tri1
$\Delta + \Delta\Delta$	tri2a
LDA+MLLT	tri2b
LDA+MLLT+MMI	tri2b_mmi
LDA+MLLT+bMMI	tri2b_bmmi
MPE	tri2b_mpe

Figure 8: Training partial order among AM in our training scripts [25]

The acoustic models mono, tri1, tri2a and tri2b are trained generatively. The models tri2b_mmi, tri2b_bmmi and tri2b_mpe are trained discriminatively in four iterations. The discriminative models yield better results than generative models if enough data is available. The discriminative models from may significantly over-fit to the training data. Discriminative training uses a unigram LM estimated on training dataset in order to compute their objective function, each iteration adapts more to the training data. We used four iterations for discriminative models training, and we have not experienced such behaviour[25].

4.3 Experiments in Voxforge database

Firs of all, we are going to analyse which is the amount of data needed to correctly train a mono-phone system (mono) and tri-phone system (tri1a). The following results presents the performance of mono-phone and tri-phone models depending on the number of utterances used to train the model.

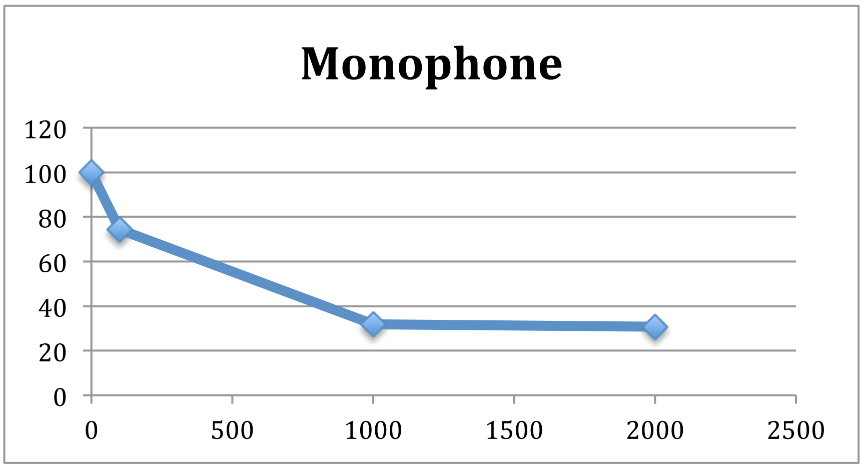


Figure 9: Displays the WER% depending the number of sentences

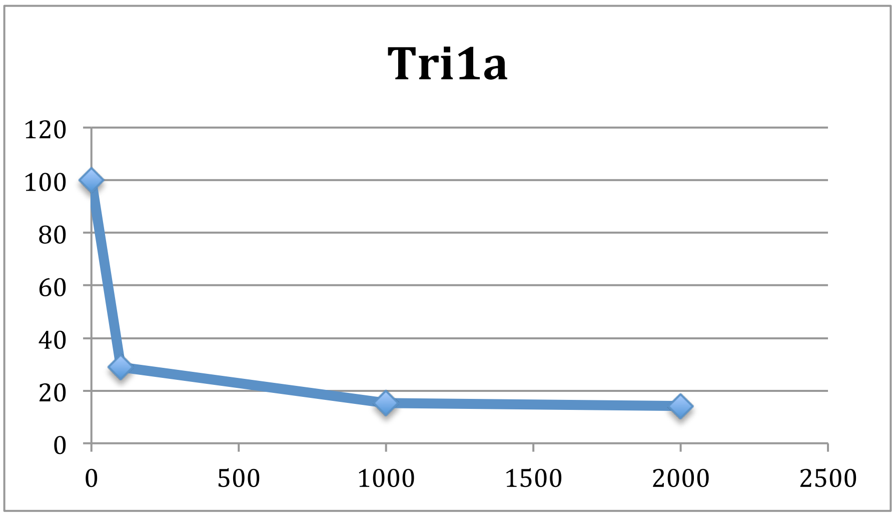


Figure 10: Displays the WER% depending the number of sentences

We can see that the best WER(%) is obtained when using triphones. Also we can see that as more training data is used, better results are achieved.

4.4 Results

In this section, will be shown the results while using different methods with the full database as training data.

Model	WER(%)
Mono	31,86
Tri1	15,22
Tri2a	16,08
Tri2b	14,93
tri2b_mmi	10,89
tri2b_mmi_b0.05	10,91
tri2b_mpe	11,61
Tri3b	12,31

Table 2: WER% of different acoustic models evaluated

The models mono, tri1 and tri2a use the $\Delta+\Delta\Delta$. We can see that mono only uses monophones and tri1 and tri2a use triphones.

Note that the tri2b uses LDA+MLLT, tri2b_mmi uses LDA+MLLT+MMI, tri2b_mmi_b0.5 uses LDA+MLLT+bMMI, and tri2b_mpe use LDA+MLLT+MPE.

From the previous results we can observe that using different linear non-dependent transforms leads to a considerably reduction of word error rate. It can be seen that LDA+MLLT works better than $\Delta+\Delta\Delta$.

4.5 Experiments in DNN

The following table presents the WER(%) trained in DNN.

Model	**Num-hidden layers**	WER(%)
Tri2a	2	8,15
	4	8,10
Tri2b	2	7,71
	4	7,68
Tri3b	5	10,63

Table 3: DNN results with WER%

We can also observe that despite increasing the number of hidden layers we do not get a significantly higher result, and the computational time is increased.
Finally, I used the script of professor Follonosa to obtain different results with the model tri3b. This model uses nnet2, but with 5 hidden layers.

4.6 Decoder

Use of multiple models in GMM-based online decoding

In the online decoding decode for GMMs in online-gmm-decoding.h, up to three models can be supplied. These are held in class OnlineGmmDecodingModels, which takes care of the logic necessary to decide which model to use for different purposes if fewer models are supplied[4].

Neural net based online decoding with iVectors

Our recommended online-decoding setup, which provides the best performance, is the neural net based setup. The adaptation philosphy is to give the neural net un-adapted and non-mean-normalized features (MFCCs, in our example recipes), and also give it an iVector. An iVector is a vector of several hundred dimensions (one or two hundred, in this particular context) which represents the speaker properties. For more information on this, the reader can look at the speaker identification literature. Our main idea is that the iVector gives the neural net as much as it needs to know about the speaker properties. This has been proved being quite useful. The iVector is estimated in a left-to-right way, meaning that at a certain time t, it sees input from time zero to t. It also sees information from previous utterances of the current speaker, if available. The iVector estimation is Maximum Likelihood, involving Gaussian Mixture Models[3].

This is used to decode the application you will see below:

```
# decoding
~/kaldi-trunk/egs/voxforge/s5/online2-wav-nnet2-latgen-faster --do-endpointing=false \
    --online=false \
    --config=nnet_a_gpu_online/conf/online_nnet2_decoding.conf \
    --max-active=7000 --beam=15.0 --lattice-beam=6.0 \
    --acoustic-scale=0.1 --word-symbol-table=graph/words.txt \
   nnet_a_gpu_online/smbr_epoch2.mdl graph/HCLG.fst "ark:echo utterance-id1 utterance-id1|"
"scp:echo utterance-id1 a0234.wav|" \
   ark:/dev/null
```

4.7 Graphical User Interface (GUI)

In this section we can see the following images that refer to the GUI programmed in python use of the PyAudio, to decode a voice audio already recorded. The set parameters are the following:

. Format = pyaudio.paInt16

. Channels = 1

. Chunk = 1024

. Rate = 16000

The program started two buttons where you can choose if you want to decode by DNN or fMLLR. By selecting one of the options, a window appears where you can select audio file desired.

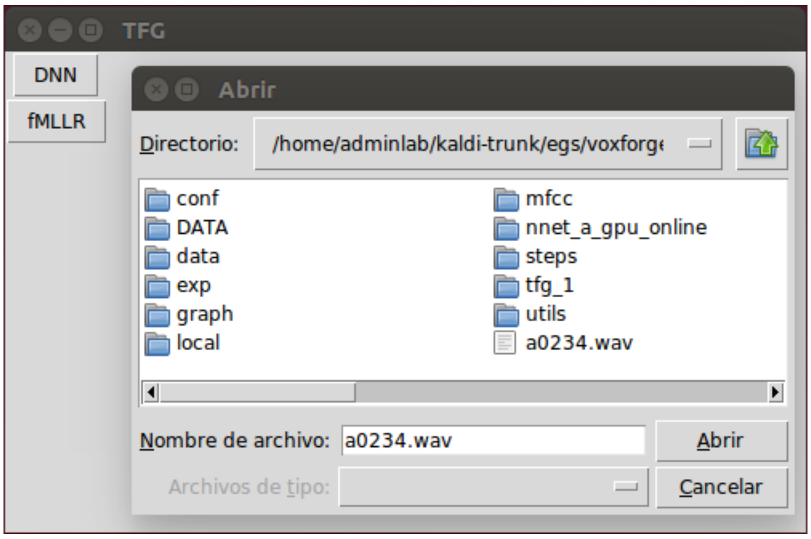


Figure 10:GUI(I)

Then, another window opens with the graphical representation of the audio voice. In order to make a better analysis of the audio file.

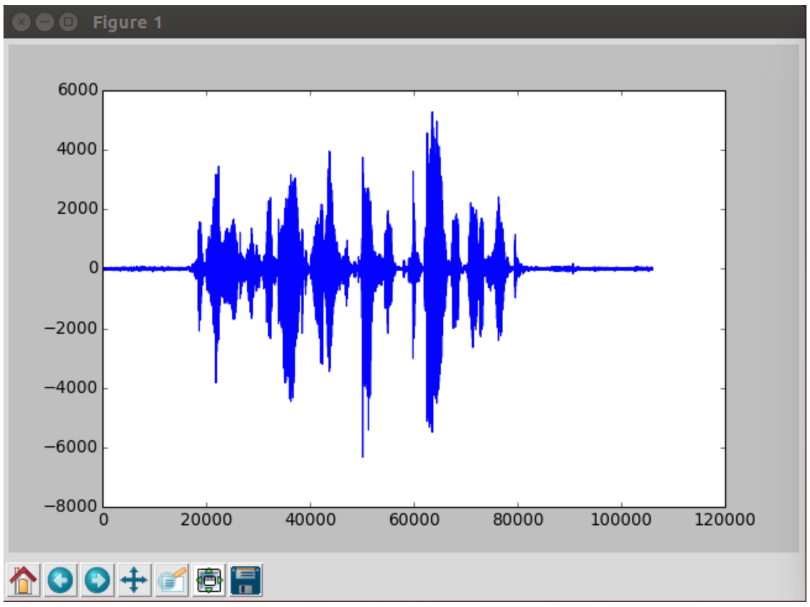
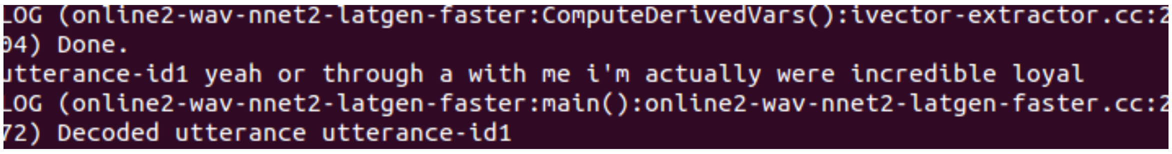


Figure 12: GUI(II)

Finally we see in the terminal decoder



```
LOG (online2-wav-nnet2-latgen-faster:ComputeDerivedVars():ivector-extractor.cc:2
04) Done.
utterance-id1 yeah or through a with me i'm actually were incredible loyal
LOG (online2-wav-nnet2-latgen-faster:main():online2-wav-nnet2-latgen-faster.cc:2
72) Decoded utterance utterance-id1
```

Figure 11: GUI(III)

5 Conclusions:

As a way of ending this project I will expose a brief and concise conclusions, deducted from the experiences and results of this thesis.

Nowadays, one of the systems of human communication with machines it's through speech recognition or ASR (automatic speech recognition). Among all the different methods that exist, this work is mainly focused in the use of new DNN. Moreover, ASR is becoming widely used, especially in the daily life of people.

Deep Neural Networks (DNN) are based in neuronal network and the use of hidden states. This allows working with better and faster decisions inside a database already trained. We have seen different methods of DNN. As an example, using triphone helps us to obtain better results. In this project we used the library kaldi, which is very versatile and usually provides good results.

From our results can be seen that DNN with lots of hidden layers are hard to optimize, therefore adding more hidden layers does not significantly improve the performance. This might be caused due to the fact that the database is no big enough to work with a high number of hidden layers

Nevertheless, it's important to remind that Deep Learning, and specifically DNN, is a very recent field. It is a really active field of research and new methods are constantly presented.

One main aspect for obtaining a good performance on an ASR system is having a database big enough, whether using DNN or finite states.

The pre-training is much more helpful in deep neural nets than in shallow ones, especially when limited amounts of labeled training data are available. Reductions in training time can be achieved with less effort by careful choice of the scales of the initial random weights in each layer.

Currently, the biggest disadvantage of DNN compared with GMM is that it is much harder to make good use of large cluster machines to train them on big datasets. This is offset by the fact that DNN make more efficient use of data so they do not require as much data to achieve the same performance.

Bibliography:

[1] Sandrine Revaz. *Statistical Models in Automatic Speech Recognition,* June 2015, University of Friboug, Switzerland.

[2] https://en.wikipedia.org/wiki/Bayes%27_theorem/

[3] Wolfgang Macherey, Discriminative Training and Acoustic Modeling for Automatic Speech Recognition, 2010.

[4] http://kaldi.sourceforge.net/

[5] Yu Hongzhi, A Research on Recognition of Tibetan Speakers Based on MFCC and Delta Features. Proceedings IFCSTA. IEEE, 2009, volume 2: pp, 234-238.

[6] R. Haeb-Umbach and H. Ney, Linear discriminant analysis for improved large vocabulary continuous speech recognition, Proceedings ICASSP. IEEE, 1992, pp, 13-16.

[7] Omar, M.K. and Hasegawa-Johnson, M. Model enforcement: a unified feature transformation framework for classification and recognition, Signal Processing, IEEE, 2004, volume 52: pp,2701-2710.

[8] D.Povey and A. Ghoshal, The Kaldi Speech Recognition Toolkit, ASRU 2011.

[9] http://www.voxforge.org/

[10] Deep Geoffrey Hinton, Li Deng and Dong Yu. Neural Networks for Acoustic Modeling in Speech Recognition, April 2012.

[11] J. P. Wilkinson, "Nonlinear resonant circuit devices," U.S. Patent 3 624 125, July 16, 1990.

[12] M.V. Alvarez Fernández. "Feasibility study and design for Wireless Sensor Networks in a space environment". M.S. thesis, Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, Delft, The Netherlands, 2011.

[13] J. O. Williams, "Narrow-band analyzer," Ph.D. dissertation, Department of Electrical Engineering, Harvard University, Cambridge, MA, USA,1993.

[14] H. Bourlard and N. Morgan, Connectionist Speech Recognition: A Hybrid Approach, Kluwer Academic Publishers, Norwell, MA, USA, 1993.

[15] Y. Hifny and S. Renals, "Speech recognition using augmented conditional random fields," IEEE Transactions on Audio, Speech & Language Processing, vol. 17, no. 2, pp. 354–365, 2009.

[16] A. Robinson, "An application to recurrent nets to phone probability estimation," IEEE Transactions on Neural Networks, vol. 5, no. 2, pp. 298–305, 1994

[17] J. Ming and F. J. Smith, "Improved phone recognition using bayesian triphone models," in Proc. ICASSP, 1998, p. 409412

[18] L. Deng and D. Yu, "Use of differential cepstra as acoustic features in hidden trajectory modelling for phonetic recognition," in Proc. ICASSP, 2007, pp. 445–448.

[19] A. Halberstadt and J. Glass, "Heterogeneous measurements and multiple classifiers for speech recognition," in Proc. ICSLP, 1998.

[20] A. Mohamed, D. Yu, and L. Deng, "Investigation of full-sequence training of deep belief networks for speech recognition," in Proceedings of Interspeech, 2010.

[21] T.N. Sainath, B. Ramabhadran, M. Picheny, D. Nahamoo, and D. Kanevsky, “Exemplar-based sparse representation features: From timit to lvcsr,” Audio, Speech, and Language Processing, IEEE Transactions on, vol. 19, no. 8, pp. 2598 –2613, nov. 2011.

[22] Y-L Chow, Maximum mutual information estimation of HMM parameters for continuous speech recognition using the< e1> N-best algorithm, Acoustics, Speech, and Signal Processing, 1990. ICASSP-90., 1990 International Conference on, IEEE, 1990, pp. 701–704.

[23] Daniel Povey, Mark JF Gales, Do Yeong Kim, and Philip C Woodland, MMI-MAP and MPE-MAP for acoustic model adaptation., INTERSPEECH, 2003.

[24] Daniel Povey, Dimitri Kanevsky, Brian Kingsbury, Bhuvana Ramabhadran, George Saon, and Karthik Visweswariah, Boosted MMI for model and feature-space discriminative training, Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on, IEEE, 2008, pp. 4057–4060.

[25] Ondřej Plátek, Speech recognition using KALDI, Prague 2014

[26] Haofeng Kou and Weijia Shang, Parallelized Feature Extraction and Acoustic Model Training, Digital Signal Processing. Proceedings ICDSP, IEEE, 2014.

Glossary

ASR: Automatic Speech Recognition

HMM: Hidden Markov Models

DNN: Deep Neural Networks

AM: Acoustic Model

LM:Language Model

GUI: Graphical User Interface

MFCC: Mel Frequency Cepstral Coefficients

PLP: Perceptual Linear Prediction

DBN: Deep Belief Net

MLLT: Maximum Likelihood Linear Transform

LDA: Linear Discriminant Analysis

RBM: Restricted Boltzmann Machine

HLDA: Heteroscedastic Linear Discriminant Analysis

MLLR: Maximum Likelihood Linear Transform

GMM: Gaussian Mixture Model

SGMM: Subspace Gaussian Mixture Model

WER: Word Error Rate

MLE: Maximum Likelihood Estimation

EM: Expectation-Maximization

MMI: Maximum Mutual Information

bMMI: Bosted Maximum Mutual Information