UNIVERSITAT POLITÈCNICA DE VALÈNCIA
DEPARTAMENT DE SISTEMES INFORMÀTICS I COMPUTACIÓ

MASTER'S THESIS

Development and Evaluation of an Automatic Speech Recognition System Adapted to the Transcription of Classroom Video Recordings

Master's Degree in Artificial Intelligence, Pattern Recognition and Digital Imaging
Academic Course 2020/2021

Nahuel Unai Roselló Beneitez

Advisers:
Dr. Albert Sanchis Navarro
Dr. Adrià Giménez Pastor

Abstract / Resum / Resumen

Abstract

Automatic Speech Recognition (ASR) has proven to be an efficient and effective way of converting speech to text over the last years. This work, performed in the context of two projects from the Government of Spain and the Generalitat Valenciana, explores the usage of ASR in the context of classroom video recordings. In order to do this, a dataset consisting of more than 1400 hours of classroom recordings is exploited. The dataset is divided into two sources (clip-on and camera microphones) which record a given class at the same time, even though one of them is noisier than the other. Several obstacles faced in the work carried out are described, such as the fact that the transcriptions of the recordings were not initially included in the dataset, or the fact that both sources of audio were not perfectly synchronized in some recordings. This work also presents experiments performed with the cleaner source of audio and replicated with both sources of audio so as to compare both approaches. Moreover, a baseline system trained with nearly 4000 hours is retrained with both sources of audio and the resulting system is compared to the rest of the developed systems. Finally, this work ends with some conclusions extracted from the previously mentioned experiments.

Resum

En els últims anys, s'ha demostrat que el Reconeixement Automàtic de la Parla (RAP) és una tècnica efectiva i eficient per convertir parla a text. Aquest treball, desenvolupat en el context de dos projectes recolzats pel Govern d'Espanya i la Generalitat Valenciana, explora l'ús del RAP en el context d'enregistraments de classes d'aula. Amb aquesta finalitat, s'explota un conjunt de dades amb més de 1400 hores d'enregistraments de classes. Aquest conjunt es composa de dos fonts de dades (micròfons de solapa i càmera) que enregistren una classe donada al mateix temps, encara que una font d'àudio té pitjor qualitat que l'altra. En aquesta memòria es descriuen alguns dels problemes que s'han donat en el treball realitzat, com el fet que inicialment el conjunt de dades no tenia cap transcripció, o que ambdues fonts d'àudio no estaven perfectament sincronitzades en alguns casos. Aquest treball també presenta experiments duts a terme amb la font de dades de millor qualitat, i replicats amb ambdues fonts d'àudio amb el fi de comparar les aproximacions. Així mateix, es reentrena un sistema ja existent amb ambdues fonts d'àudio. El sistema resultant, entrenat prèviament amb quasi 4000 hores d'àudio, es compara a la resta dels sistemes desenvolupats. Finalment, aquest treball exposa algunes conclusions extretes dels experiments anteriorment esmentats.

Resumen

El Reconocimiento Automático del Habla (RAH) ha demostrado ser una manera efectiva y eficiente de convertir habla a texto a lo largo de los últimos años. Este trabajo, desarrollado en el contexto de dos proyectos apoyados por el Gobierno de España y la Generalitat Valenciana, explora el uso del RAH en el contexto de grabaciones de clases de aula. Con este fin, se explota un conjunto de datos con más de 1400 horas de grabaciones de clases. Este conjunto se compone de dos fuentes de datos (micrófonos de solapa y cámara) que graban una clase determinada al mismo tiempo, aunque una de las fuentes tiene peor calidad que la otra. A lo largo de esta memoria, se describen algunos de los problemas que se han dado en los proyectos, como el hecho de que inicialmente el conjunto de datos no viene dado con ninguna transcripción, o que ambas fuentes de datos no estaban perfectamente sincronizadas en algunos casos. Este trabajo también presenta experimentos llevados a cabo con la fuente de datos de mejor calidad, y replicados con ambas fuentes de audio con el fin de comparar las dos aproximaciones. Además, se reentrena un sistema ya existente con ambas fuentes de audio. El sistema resultante, previamente entrenado con casi 4000 horas de audio, se compara con el resto de sistemas desarrollados. Finalmente, este trabajo expone algunas conclusiones extraídas de los experimentos anteriormente mencionados.

Contents

Abstract / Resum / Resumen **iii**

1 Introduction **1**
1.1 Motivation . . . 1
1.2 Main objectives . . . 2
1.3 Document structure . . . 2

2 Fundamentals of Automatic Speech Recognition **5**
2.1 Introduction . . . 5
2.2 Neural networks . . . 7
2.3 State-of-the-art neural network architectures . . . 8
2.3.1 Recurrent neural networks . . . 9
2.3.2 Other neural network architectures . . . 10
2.4 Statistical ASR . . . 12
2.5 Acoustic preprocessing . . . 13
2.6 Data augmentation . . . 16
2.7 Acoustic modeling: Hidden Markov Models . . . 17
2.8 Acoustic modeling: DNN-HMM hybrid approach . . . 20
2.8.1 Recurrent-based acoustic modeling . . . 20
2.8.2 Other acoustic model types . . . 20
2.9 Decoding . . . 21
2.10 ASR Evaluation . . . 24

3 Project description, dataset and tools used **25**
3.1 Multisub and CAR research projects . . . 25
3.1.1 Multisub . . . 25
3.1.2 Classroom Activity Recognition . . . 26
3.2 Videoapuntes dataset description . . . 27
3.3 Tools used . . . 29
3.3.1 TLK . . . 29
3.3.2 TensorFlow . . . 30

4 Acoustic model training **31**
4.1 Introduction . . . 31
4.2 Baseline system . . . 32
4.3 Training pipeline . . . 33
4.3.1 Generation of unsupervised transcriptions with clip-on audio . 34
4.3.2 Alignment of the unsupervised transcriptions . . . 34

4.4 Features of the developed acoustic models 34
4.5 First training strategy: training from scratch 36
4.5.1 Clip-on audio system 36
4.5.2 Clip-on and camera audio system 36
4.6 Second training strategy: fine-tuning of the baseline MLLP-UPV Spanish ASR system 37

5 ASR system evaluation **41**
5.1 Introduction 41
5.2 Parameter optimization 42
5.3 Senone prior extraction 43
5.4 Evaluation of the clip-on ASR system trained from scratch 43
5.5 Evaluation of the clip-on and camera audio system trained from scratch 46
5.6 Evaluation of the fine-tuned ASR system 48
5.7 Discussion 50

6 Conclusions **53**
6.1 Future Work 54

Acknowledgements **55**

Chapter 1
Introduction

This work explores the development of Automatic Speech Recognition (ASR) systems with the main aim of transcribing real lessons lectured at the Universitat Politècnica de València (UPV) in the Spanish language.

In this chapter, the main motivation is described, and the core objectives of the work are specified. Moreover, the reader will find an outline of the document at the end of this chapter.

1.1 Motivation

The need of this Master's Thesis is motivated by two projects: the **Multilingual subtitling of classrooms and plenary sessions** (**Multisub**) [1] project, a 3-year project selected for funding by the Ministerio de Ciencia, Innovación y Universidades and addressed by the *Machine Learning and Language Processing - Universitat Politècnica de València* (MLLP-UPV) research group, and the **Classroom Activity Recognition** (CAR) project, a 4-year project funded by the Conselleria d'Educació, Investigació, Cultura i Esport de la Generalitat Valenciana and addressed by the Valencian Research Institute for Artificial Intelligence (VRAIN). These projects will be thoroughly described in Chapter 3, but an outline is given below.

The **Multisub** project aims at modernizing educational resources by including both lecture recordings as a support tool for both teachers and students, as well as parliamentary resources by making data accessible to the general public. On the other hand, the objective of the **CAR** project is to analyze the different interactions between a teacher and their students, building a model of student engagement as well as a model of teaching methodology.

Most students can benefit from the transcription of online classes in order to have available the lecture in text format, supporting the lecturer's explanation and any other auxiliary material available by the student. Moreover, thanks to these transcriptions, other students in special conditions such as foreign or disabled students could be able to access the data in a way that is readable for them: for instance, a student that does not understand the Spanish language could have these sentences translated to their native language by means of a translation system, while a deaf person could immediately access the information transmitted by the lecturer.

Educational resources are shifting their attention to a mixed methodology which has been recently put to test by the COVID-19 pandemic. This situation has required social distancing and, in certain occasions, attending classes in a strictly online format. In this regard, the recording of online lectures has made available a vast amount of data that can be taken advantage of in order to develop high-quality ASR systems, which is the intended purpose of this work.

Moreover, as it will be detailed in Chapter 3, the data is split into two sources of data: audio coming from a clip-on microphone, and audio coming from a camera microphone. Both microphones should record the same lecture at the same time. Even though the clip-on microphone yields audio which is cleaner than the camera microphone, it is desirable to exploit both sources of data. Furthermore, most of the data is unlabeled, which is why this Master's Thesis will also explore semi-supervised learning.

Finally, it is essential that the developed systems achieve an acceptable error rate in the audio coming from the camera microphone, since it is the most common source of audio in real online classes due to the fact that the clip-on microphone is not available in most real lectures. This is in some way intended because the camera microphone works as a far-field microphone which is able to capture, for instance, students' interventions, while the clip-on microphone can only capture the lecturer's voice.

1.2 Main objectives

The main objectives of this work are the following:

- To gain a deeper understanding of state-of-the-art techniques applied to the field of speech recognition.
- To explore different training strategies using information coming from two sources (clip-on and camera microphones).
- To develop several ASR systems that are able to produce acceptable automatic transcriptions from classroom video recordings.
- To apply the developed ASR systems for transcribing classroom video recordings at the Universitat Politècnica de València (UPV) in the context of the **Multisub** and **CAR** projects.

1.3 Document structure

This document is structured in the following way:

- Chapter 1 has established the main motivation for developing this Master's Thesis and has introduced the main objectives to be fulfilled in this work.

- Chapter 2 will introduce the fundamental knowledge that every reader should acquire before going deeper into the work, while also introducing state-of-the-art architectures used in ASR.
- Chapter 3 describes the projects in which this work is encapsulated, and introduces both the tools used and the main dataset: a collection of data coming from online classes lectured at the UPV.
- Chapter 4 describes the work done in order to train the main systems developed in this work.
- Chapter 5 reports the results obtained in the evaluation process of the different systems trained.
- Finally, Chapter 6 presents some conclusions obtained from the work performed in this Master's Thesis.

Chapter 2

Fundamentals of Automatic Speech Recognition

In this chapter, the essential knowledge in order to understand the ASR field is provided. This chapter will begin by giving general notions as an introduction. Two sections will then be devoted to explaining the foundations of neural networks and state-of-the-art neural network architectures, while later on going deeper into the formal foundations of ASR. After that, the process of preprocessing acoustic samples will be described, data augmentation in the context of ASR will be introduced, and state-of-the-art ASR models will be detailed. Finally, evaluation measurements for ASR systems will be provided.

2.1 Introduction

ASR could be defined as the process that obtains the most probable sequence of words according to a certain acoustic signal. It is a cheap but effective way of obtaining the text which has been uttered in a given recording. Many researchers and commercial companies have put a vast amount of effort into this field, which is one of the reasons why ASR is one of the most popular fields of machine learning nowadays.

In the past, ASR was behind Human Speech Recognition (HSR) in terms of error rate. A study performed in 1997 found that, in spontaneous speech over the Switchboard dataset [13], the error rate of a machine was located at a 43% while its human counterpart was of a 4% [26]. However, since then the field has evolved and more powerful methods have emerged. The introduction of techniques such as Artificial Neural Networks (ANNs), and more recently Deep Neural Networks (DNNs; they will be described in the next section) to ASR have boosted the recognition rate of the most recent systems. Newer studies on the effectiveness of ASR show that automatic recognition of utterances is approaching human accuracy. In fact, recent articles set the error rate on the Switchboard dataset to 5.9 error points in 2017 [46] and to 5 error points in 2021, "practically reaching the limit of the benchmark" [47].

One of the fields which could benefit the most from ASR is healthcare. Besides the huge benefits for people with hearing loss, ASR can help in the diagnostic, treatment or everyday life of patients while supporting professionals. In [59], a person's narrative spontaneous speech was transcribed and later processed in order to classify the person as being healthy or suffering Alzheimer's disease. In [40], patients were asked to provide a description of a picture, being proposed a similar approach to the previous one. Both approaches would otherwise require valuable resources, such as professional time, expensive scans, or blood or spinal samples [40].

Moreover, ASR can also assist in the recognition of dysarthric speech[1], as seen in [54], where a spoken digit classification task with dysarthric patients was addressed by using ASR techniques, or in [44], where the RGB spectrogram of single words was analyzed in order to recognize uttered by dysarthric individuals. Successful use cases of ASR in the health domain include MobileCogniTracker [52], a mobile application in charge of gathering remote cognitive tests. This application integrates speech-to-text functionality, which allows the test to resemble the traditional one while helping people with reduced motor coordination.

Another area which is in demand as of today is the area of virtual assistants. Many commercial companies have created their own virtual assistant, which is able to interact with the end user in order to fulfill their needs. These technologies heavily rely on ASR to make the overall user experience as comfortable as possible. Some examples include Alexa (Amazon)[2], Siri (Apple)[3], Bixby (Samsung)[4], or Assistant (Google)[5].

One last area of application that is worth mentioning is the generation of automatic transcriptions in video repositories. These repositories can be academic, cultural, scientific, political... In these huge repositories, it is interesting to have available an ASR system that yields acceptable transcriptions which can then be postprocessed by experts, or even immediately used as the video transcription. In the case of universities, they are beginning to see benefits in recording classroom lectures to support learning. From Massive Open Online Courses (MOOCs) to usual lectures, the usage of online learning platforms is being popularized among students, who use these platforms for reviewing concepts, attending the classes that they missed or taking additional notes.

Thanks to the effort made on building open toolkits capable of developing ASR systems such as Kaldi [33] or HTK [56], as well as open audio datasets like Mozilla Common Voice [4], Librispeech [31], TED-LIUM [38], Europarl-ASR [12], or VoxPopuli [49], nowadays speech recognition can be accessed by everyone. Readers interested on more information about available datasets are pointed to [19].

[1]Dysarthric speech is caused by dysarthria, a disease that affects the motor system and is characterized by a reduction in the intelligibility of the patient's speech [54].

[2]`https://developer.amazon.com/en-US/alexa`. Last accessed: 11-09-2021.

[3]`https://www.apple.com/siri/`. Last accessed: 11-09-2021.

[4]`https://www.samsung.com/us/explore/bixby/`. Last accessed: 11-09-2021.

[5]`https://assistant.google.com/`. Last accessed: 11-09-2021.

2.2 Neural networks

Before introducing ASR systems, it is essential to learn about neural networks, since they will be frequently mentioned throughout this work. A Neural Network (NN), also called connectionist model, Multilayered Perceptron (MLP) or Artificial Neural Network (ANN), is a set of densely interconnected simple processors, also called neurons. These processors are usually grouped in layers, where all processors of a layer are connected to all processors of the following layer. A neural network has at least an input layer (where the input is received), a hidden layer (where the input is further processed) and an output layer (which yields the output).

In a neural network, the information is trasmitted from the i-th neuron of the layer $(k-1)$ to the j-th neuron of the layer k according to the following formula:

$$s_j^k = g(\sum_{i=0}^{M_{k-1}} \theta_{ji}^k \cdot s_i^{k-1}), 1 \leq j \leq M_k \tag{2.1}$$

where s_j^k is the output[6] of the neuron j, θ_{ji}^k is a weight connecting the i-th neuron to the j-th neuron, M_k is the number of neurons in the layer k, and $g(\cdot)$ is a non-linear function, also called activation function.

This basic model of neural network, where there are no cycles and information can only flow forward, is also called Feed-Forward Neural Network (FFNN). Figure 2.1 shows a generic (feed-foward) neural network.

In this work only four activation functions will be described, but the reader is pointed to [30] in order to check the different activation functions and their applications.

The first three activation functions to be described are the ReLU, the sigmoid and the hyperbolic tangent functions, respectively defined as:

$$\text{ReLU}(z_i) = max(0, z_i) \tag{2.2}$$

$$\sigma(z_i) = \frac{1}{1 + \exp(-z_i)} \tag{2.3}$$

$$\tanh(z_i) = \frac{\exp(z_i) - \exp(-z_i)}{\exp(z_i) + \exp(-z_i)} \tag{2.4}$$

where $\exp(x) = e^x$. Nowadays, it is common to use the ReLU function as the activation function of the hidden layer, but the other two are also used in some cases. The last activation function which will be mentioned is the softmax function, defined over a vector z, where each output is defined as:

$$\text{softmax}(z_i) = \frac{\exp(z_i)}{\sum_j \exp(z_j)} \tag{2.5}$$

[6]In the particular case of $k = 0$, in order not to lose generality it can be considered that $s_i^0 = x_i$, where x is the input to the neural network.

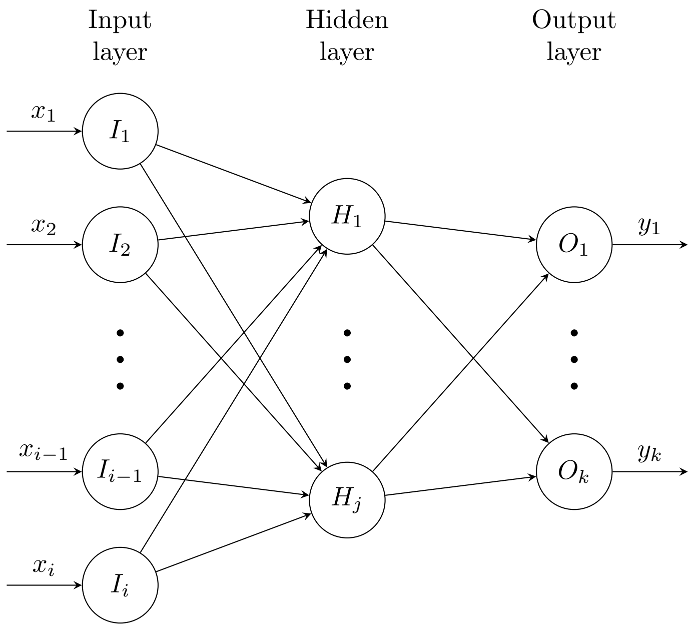


Figure 2.1: General structure of a neural network. In this neural network, $M_0 = i$, $M_1 = j$ and $M_2 = k$. Each neuron m is connected with every other neuron n of the following layer by a weight, which is usually represented as θ_{nm}. The biases of the network, which provide a fixed coefficient of $+1$, are not included in this figure.

This function defines a probability distribution among all elements of z. As a consequence, it is generally used as the activation function of the output layer, taking advantage of the discriminative power of neural networks.

The set of network weights θ can be learned through the backpropagation algorithm. In this algorithm, each sample is processed until the output layer. The derivative of the error is then brought back until the input layer. Finally, a new set of weights θ' can be calculated, modifying each weight proportionally to the error obtained. For a formal description of the algorithm, the reader is addressed to [39].

A neural network can have more than one hidden layer (usually more than two). In this case, it is called a Deep Neural Network (DNN). This allows for a better sample representation, since DNNs learn a feature hierarchy: the hidden layers which are closer to the input layer represent low-level, local features, while the hidden layers closer to the output (usually softmax) layer represent high-level, invariant features which help the classification process [57].

2.3 State-of-the-art neural network architectures

Nowadays, there are many types of neural networks. In this section, the neural network architecture used in this work will be described, the Recurrent Neural Network (RNN). Moreover, other interesting types of neural networks will also be outlined.

2.3.1 Recurrent neural networks

Recurrence is defined as "the fact of happening again"[7]. This definition implies a temporal sequence, and dependencies among periods of time. A Recurrent Neural Network (RNN) is able to capture this information and the underlying dependencies thanks to its recurrent connections, that is, connections among neurons that form a directed cycle, creating a memory expressed as internal states [57]. Therefore, RNNs are well-suited to model temporal sequences, which has been shown by their excellent performance for a very long time. Near their introduction to the field of ASR, RNNs were able to improve the already established state of the art in phoneme recognition [36]. Today, this architecture is widely used in ASR.

Inside the recurrent-based architectures, the Long Short Term Memory (LSTM) architecture [17] can be found. This architecture enhances the capabilities of the RNNs by identifying the exploding or vanishing gradient problem as a source of errors of an RNN and establishing an LSTM cell as the essential unit. This cell is composed of three types of gates: input gates, in charge of controlling the flow of information entering the cell; output gates, which control the flow of information leaving the cell; and forget gates, that control whether the information inside the cell disappears or not. The cell also learns a set of weights to optimally combine these parameters [17]. Figure 2.2 shows the basic structure of an LSTM cell.

The Bidirectional LSTM (BLSTM) architecture adds bidirectionality to LSTMs, which is achieved by splitting the desired layer in two parts: one that analyzes the temporal sequence in the forward sense, and another that analyzes it in the backward sense. These are referred to as the forward and the backward states respectively. The input at time t to the BLSTM, x_t, is therefore

$$x_t = [\overrightarrow{x_t}, \overleftarrow{x_t}]$$

where $\overrightarrow{x_t}$ and $\overleftarrow{x_t}$ are the result (at time t) of analyzing the sequence in the forward and the backward sense, respectively [42]. Figure 2.3 shows the generic bidirectional schema, while Figure 2.4 shows the common structure of a BLSTM network applied to speech recognition. In it, the division of a BLSTM layer in forward and backward neurons can be seen. Each of these groups is connected to both groups of the following layer in order to have full information from the analyzed sequence.

[7] *recurrence*. Cambridge Dictionary. `https://dictionary.cambridge.org/dictionary/english/recurrence`. Last accessed: 11-09-2021.

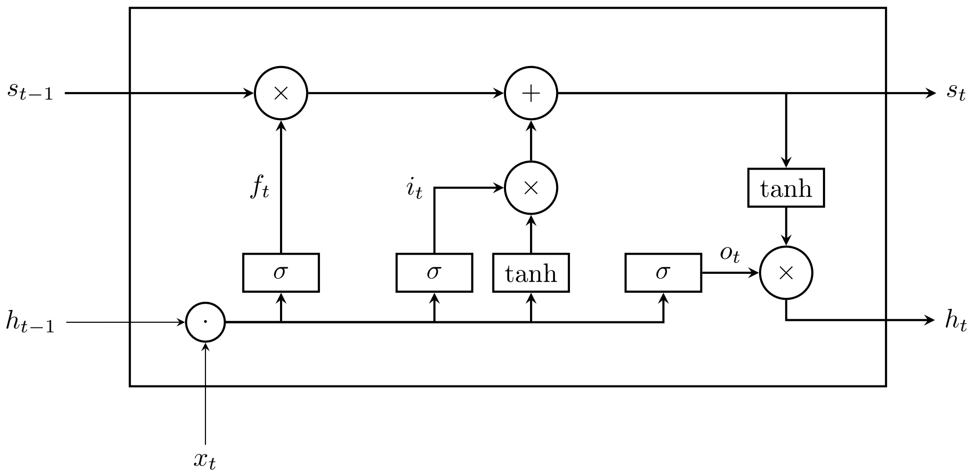
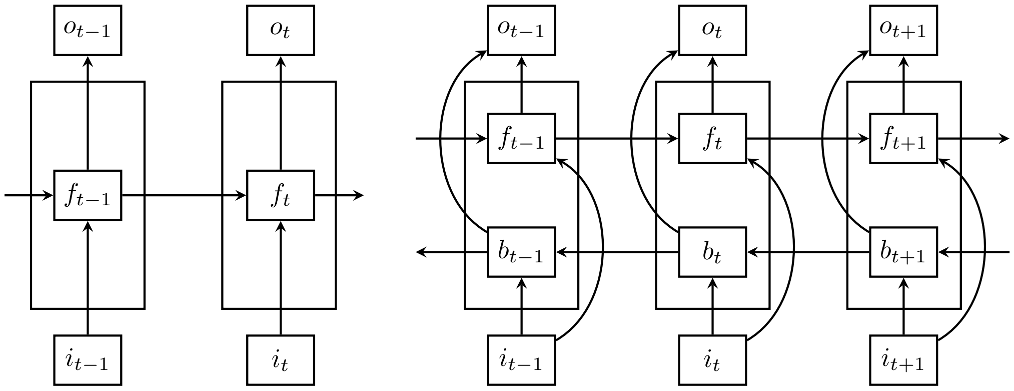


Figure 2.2: Internal view of an LSTM memory cell. In the figure, x_t, h_t and s_t are the input, the output and the cell state at time t, while f_t, i_t and o_t are the forget, input and output coefficients at time t, which regulate the flows through the LSTM. Circles with $+$, $\times$ and $\cdot$ denote the multiplication, the addition and the concatenation operations, while squares labeled with σ and "tanh" denote the sigmoid and the hyperbolic tangent activation functions, respectively.



(a) Unidirectional recurrent network **(b)** Bidirectional recurrent network

Figure 2.3: Unidirectional network (left) versus bidirectional network (right). These images correspond to an unfolded recurrent neural network. In a time instant t, f_t and b_t are the forward and the backward part of the bidirectional network, i_t is the input to the layer, and o_t is the output of the layer (both of which may be another layer).

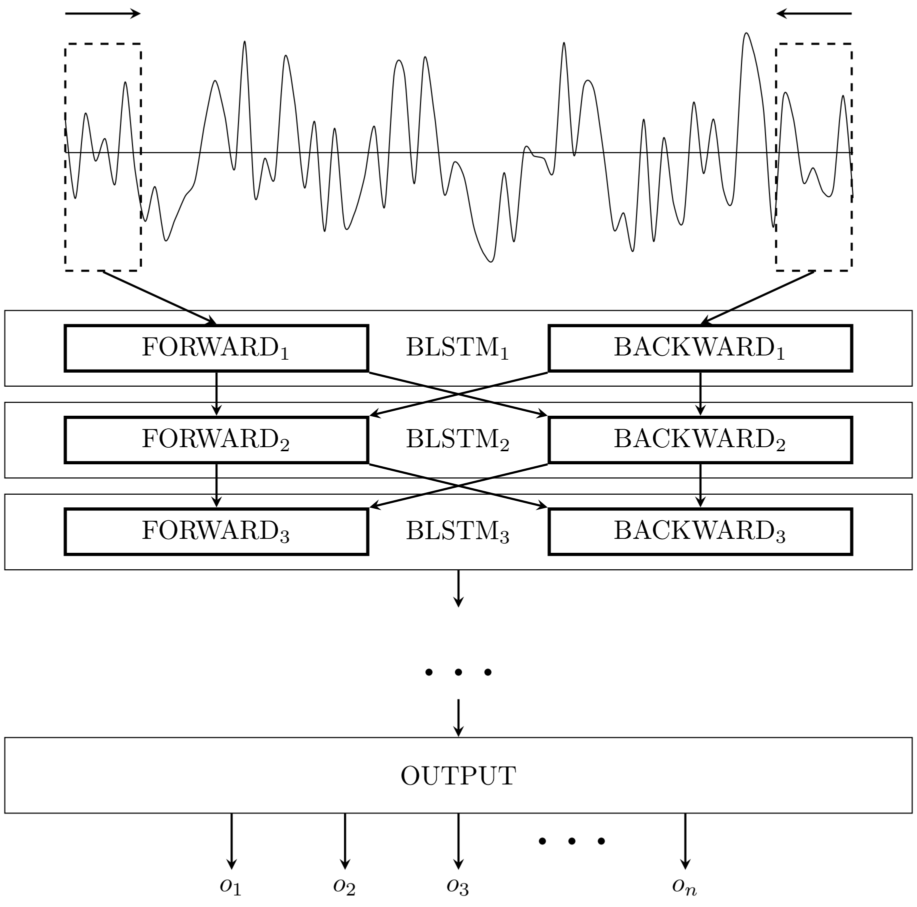


Figure 2.4: Structure of a basic BLSTM network oriented to speech recognition. The input to the network is meant to be the feature vectors obtained by the preprocessing of the windows considered in the audio, marked as dashed rectangles in the figure. Each output o_i is the probability of the i-th acoustic phenomenon taking place in the leftmost window.

2.3.2 Other neural network architectures

Besides the ones described above, there are many other types of neural network architectures. Two of the most well-known architectures are the Transformer architecture and the Convolutional Neural Network (CNN), both of which will be briefly outlined in the following:

- The Transformer architecture is a novel encoder-decoder architecture paired with the idea of attention, which allows modeling dependencies without taking into account the distance of the sequence [48]. These attention mechanisms were generally paired with an RNN. The Transformer, however, is not supported on recurrence, which allows more parallelism in the training phase. Instead, it uses self-attention (attention computed over a sequence in order to produce a representation of such sequence) in order to relate two arbitrary elements in a time which is constant with respect to the relative distance of the elements [48].

- Convolutional Neural Networks (CNNs) apply the convolution operator (element-wise multiplication) to a map with respect to a set of convolutional kernels (also called filters or simply kernels) in order to obtain a new map for each kernel used. In a CNN, these kernels are grouped in layers called convolutional layers. It is also common for convolutional layers to be followed by pooling layers, which reduce the size of the resulting maps and group information with respect to local information obtained from the map, usually the maximum or average value of the considered area. Due to their nature, they have been widely applied to image recognition.

Both of these architectures have been used in ASR. Their usage will be described in Section 2.8.2.

2.4 Statistical ASR

ASR is encapsulated inside the branch of engineering known as Pattern Recognition (PR), which deals with the automatic discovery of patterns in a usually large amount of data [7]. In these types of statistical classifiers, the Bayes' formula is usually applied. This formula relates the posterior probability of a class given a sample with the posterior probability of the sample given the class through the following equation:

$$P(c \mid x) = \frac{P(x \mid c) \cdot P(c)}{P(x)} \tag{2.6}$$

where x is the sample (usually expressed as a vector) and c is the class of the sample. When talking about classification, that is, looking for the most fitting class $\hat{c}$ to the sample x, the highest probability $P(c \mid x)$ must be found:

$$\hat{c} = \operatorname*{argmax}_{c \in C} P(c \mid x) = \operatorname*{argmax}_{c \in C} \frac{P(x \mid c) \cdot P(c)}{P(x)} = \operatorname*{argmax}_{c \in C} P(x \mid c) \cdot P(c) \tag{2.7}$$

where C is the set of classes of the problem. The denominator $P(x)$ can be safely dropped since a maximization with respect to c is taking place, so $P(x)$ will be a constant and thus irrelevant to the maximization.

The challenge that comes with PR systems which rely on Bayes' formula is the difficulty of estimating both $P(x \mid c)$ and $P(c)$. These probabilities must be approximated, since the number of possible inputs that the system can receive is potentially infinite. In this case, a model is defined. This model should allow capturing features of the problem at hand, learning patterns of the seen samples that allow their correct classification while learning to distinguish among the different classes for handling potentially unseen data.

In the field of ASR, the sample x is usually substituted with a sequence of acoustic vectors $a_1^n = a_1 a_2 \ldots a_n$ that represent the audio sample, while the class c is normally substituted with a sequence of words $w_1^m = w_1 w_2 \ldots w_m$. The substitution of the relevant terms in Equation 2.7 yields as a result the following optimization:

$$\hat{w} = \underset{w_1^m \in W^*}{\operatorname{argmax}} P(w_1^m \mid a_1^n) = \underset{w_1^m \in W^*}{\operatorname{argmax}} P(a_1^n \mid w_1^m) \cdot P(w_1^m) \tag{2.8}$$

where W^* is the set of possible sentences in a language. It is also usual to work with the corresponding log-probabilities:

$$\hat{w} = \underset{w_1^m \in W^*}{\operatorname{argmax}} \log(P(a_1^n \mid w_1^m) \cdot P(w_1^m)) = \underset{w_1^m \in W^*}{\operatorname{argmax}} \log P(a_1^n \mid w_1^m) + \log P(w_1^m) \tag{2.9}$$

The model that estimates the probability $P(a_1^n \mid w_1^m)$ is referred to as the *acoustic model*, while the model that estimates the probability $P(w_1^m)$ is called the *language model*. This work will mostly focus on extracting different acoustic models based on BLSTMs, while the language model will be given.

2.5 Acoustic preprocessing

As with many other types of systems, ASR systems do not usually work with raw data. Prior to being able to train the system and obtain the transcriptions of a given audio sample, the acoustic data needs to be preprocessed. This procedure tries to capture the most relevant features of the acoustic sample, and yields as a result a sequence of acoustic features referred to as a_1^n in the previous section. In the following, the process of extracting an arbitrary acoustic vector a_i will be explained [11] [23][8].

In order to follow more easily the process as a whole, the notation of a vector will include as a subscript the (usually three) first letters of the process by which it was extracted; the vector's i-th position will be declared between parentheses. For instance, $x_{\text{raw}}(n)$ would denote the n-th element of the raw audio. Moreover, the multiplication will be explicitly stated through the $\cdot$ operator most of the time, except in obvious cases such as 2π.

The first step is to split the audio stream into equally-sized windows. The size of these windows should be on the order of milliseconds, big enough to capture a speech sound but small enough so that at most one speech sound is captured. Common measures for obtaining the windows are 25 ms of window length, and 10 ms of shift from one window to another. This creates a sequence of overlapped windows, each of which will be eventually converted to the desired acoustic vector a_i.

The next thing to be done after obtaining the windows is to subtract the mean to each window. If this step was not performed, the frequential analysis would include additional, unimportant data in the output [16].

[8]The specific chapter which supports this section is Chapter 9. The respective information in the draft of the book's third edition can be found in Chapter 26. Such draft may be found in `https://web.stanford.edu/~jurafsky/slp3/`, which was updated on 30-12-2020. Last accessed: 11-09-2021.

After removing the mean, the next step applied is called preemphasis, and consists of applying to the waveform a formula similar to a derivative function in order to capture the most relevant changes of the audio signal, balancing the power of high frequencies with respect to low ones. The resulting audio signal x_{pre} is computed as:

$$x_{\text{pre}}(n) = x_{\text{raw}}(n) - k \cdot x_{\text{raw}}(n-1) \tag{2.10}$$

where x_{raw} is the original audio signal and $0.9 \leq k \leq 1.0$ is a coefficient which controls the factor in which the previous signal is considered. Note that this operation may need elements from the previous window, which should be provided as needed.

Since the frequential analysis through the Fourier transform requires the waveform to be periodic, it is useful that both the beginning and the end of the waveform are not abruptly cut. To do this, the sides are faded in and out by using a windowing function [16], which is applied to the waveform in the following way:

$$x_{\text{wnd}}(n) = x_{\text{pre}}(n) \cdot w(n) \tag{2.11}$$

where $w(n)$ is the chosen windowing function. There are many windowing functions, but one of the most commonly used is the Hamming function, whose corresponding $w(n)$ is the following:

$$w(n) = \begin{cases} 0.54 - 0.46 \cdot \cos(\frac{2\pi \cdot n}{L}) & 0 \leq n \leq L-1 \\ 0 & \text{otherwise} \end{cases} \tag{2.12}$$

where L is the window length. Using this window function gives rise to the Hamming window. Since the values at the edges of the function are small (which is intended in order to make the window fade in and out), windows should be overlapping, since otherwise information would then be missed during the analysis [16].

Then, the Discrete Fourier Transform (DFT) is applied to the resulting windows to convert a signal from the temporal domain into the frequential domain. The basic formula of the DFT is the following:

$$x_{\text{DFT}}(m) = \sum_{n=0}^{N-1} x_{\text{wnd}}(n) \cdot \exp\left(-2\pi \cdot \mathrm{i} \cdot \frac{m \cdot n}{N}\right), 0 \leq m \leq N-1 \tag{2.13}$$

where $\mathrm{i} = \sqrt{-1}$, $\exp(x) = e^x$ and N is the number of frequency components that need to be extracted. This formula not only obtains the frequencies and their phases, but also the magnitude of each respective frequency, which in this case corresponds to the energy or amplitude. A common and efficient way of implementing the DFT is through the Fast Fourier Transform (FFT) algorithm, whose only limitation is that N is a power of two.

With the output of the frequential analysis, the non-linearities present in the human hearing can now be modeled: lower frequencies are much better discriminated than higher frequencies. This allows for a less refined classification by placing the different frequencies in buckets, while still maintaining the general frequency information, which is done by applying mel frequency filter banks. These get their name from the mel scale, calculated as follows:

$$mel(f) = 1127 \cdot \ln\left(1 + \frac{f}{700}\right) \tag{2.14}$$

Every filter (also known as channel) of the filter bank maps to a single dimension of the final vector. Figure 2.5 shows a filter bank consisting of 16 filters.

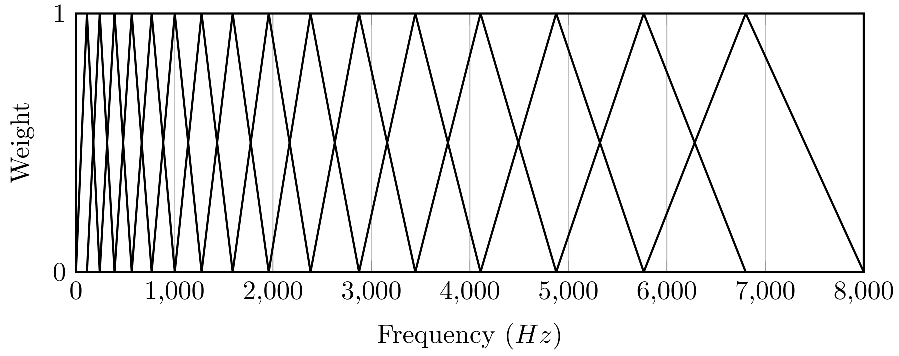


Figure 2.5: Mel filter bank consisting of 16 different filters. Each of the different filters corresponds to a triangle.

The contribution of a frequency k to each of the filters composing the filter bank H_m can be calculated in the following way:

$$H_m(k) = \begin{cases} \frac{k - f(m-1)}{f(m) - f(m-1)} & f(m-1) \leq k \leq f(m) \\ \frac{f(m+1) - k}{f(m+1) - f(m)} & f(m) < k \leq f(m+1) \\ 0 & \text{otherwise} \end{cases} \tag{2.15}$$

where m is the m-th filter, and $f(m-1)$, $f(m)$ and $f(m+1)$ are the left side, center and right side of the m-th filter, defined by the conversion of three equidistant frequencies (measured in hertz) to the mel scale. Finally, the contribution of every frequency to each element of the final vector is calculated by iterating over all frequencies:

$$x_{\text{FB}}(i) = \sum_{l=0}^{\frac{L}{2}-1} x_{\text{DFT}}(l) \cdot H_m\left(F_m \cdot \frac{l}{L}\right) \tag{2.16}$$

where F_m is the sampling frequency. Since the output of the DFT is supposed to be symmetrical, it is enough by iterating over the first half of its output.

After obtaining the mel filter bank features, they are subject to a logarithmic transformation, which also tries to model the human hearing but this time with respect to energy: lower amplitudes are discriminated better than higher amplitudes. This operation is trivially defined as:

$$x_{\log}(i) = \log(x_{\text{FB}}(i)) \tag{2.17}$$

After this step, the inverse Discrete Fourier Transform is applied. This operation transforms the frequential data back to the temporal domain, obtaining the *cepstrum*[9]. It is applied in order to decorrelate the output of the filter banks, and it is implemented through the Discrete Cosine Transform (DCT):

$$x_{\mathrm{DCT}}(i) = \sum_{j=1}^{N} x_{\log}(j) \cdot \cos\left(\frac{\pi \mathrm{i}}{N} \cdot (j - 0.5)\right) \tag{2.18}$$

where N is the number of filters in the filter bank.

Finally, both the derivative (delta) and second derivative (double delta) are computed by taking previously computed vectors as required. These deltas are appended to the vector x_{DCT}, creating a new vector x_{MFCC}. Each of the features of this new vector is called Mel Frequency Cepstral Coefficient (MFCC) feature, and the whole vector is often called MFCC vector.

The full process historically done was described for completeness. However, as of today, the logarithm of the filter bank acoustic vectors ($x_{\log}$, also called log-mel filter bank coefficient vectors) is used as input of DNNs, using many filters in order to obtain a better resolution (for instance, this work uses 85 filters when obtaining the feature vectors), in some cases adding the delta or double-delta features. Using these acoustic vectors has shown better results than converting the data back to the temporal domain and using the traditional MFCC vectors instead [57].

2.6 Data augmentation

Data augmentation refers to the set of techniques applied to the data used during the training phase that slightly change its representation so that the system does not overfit to the data in the train set. Data augmentation in ASR usually modifies the spectrogram or the audio itself.

One of the most well-known data augmentation techniques in ASR is SpecAugment [32]. This technique was proposed in 2019 and it resembles other techniques coming from different domains, such as cutout[10] or random erasing[11] in image processing. It works at the spectrogram level, directly affecting the feature vectors, and proposes three kinds of modifications:

- Moving the signal in the temporal line. This is carried out by shifting the acoustic signal for a small amount.

- Masking the signal in the frequency domain. This can be done by taking consecutive frequency channels, and applying to them the wanted masking value.

[9]The *ceps-* prefix in *cepstrum* is the inverse of the prefix *spec-* in *spectrum.*

[10]T. DeVries and G. W. Taylor. Improved Regularization of Convolutional Neural Networks with Cutout. In arXiv:1708.04552, 2017.

[11]Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang. Random Erasing Data Augmentation. In arXiv:1708.04896, 2017.

- Masking the signal in the temporal domain. This is achieved by taking consecutive time steps, and applying to them the wanted masking value.

The masking value is usually zero, which implies that no information is kept in the masking process. This process prevents overfitting and grants improvements with respect to models that do not use this data augmentation technique [32].

Another data augmentation technique is speed perturbation [25]. This technique slows down or speeds up the audio waveform by time-warping it through a parameter denoted as α in the original paper, and calculating the new signal at time t as $x_{\text{SP}}(t) = x_{\text{raw}}(\alpha t)$.

The last interesting data augmentation technique to be mentioned is noise injection [55], which artificially adds noise to an otherwise noise-free dataset. This noise can either be created or taken from another dataset specialized in background noise.

2.7 Acoustic modeling: Hidden Markov Models

This section will establish the foundations of acoustic modeling, so that the reader can fully understand the next section, which will discuss the state of the art in acoustic modeling.

As it has been mentioned in Section 2.4, the acoustic model is in charge of estimating $P(a_1^n \mid w_1^m)$, that is, the probability that a sequence of feature vectors is generated by a sequence of words. Acoustic preprocessing helps by extracting the most relevant features of the audio, but how to treat these feature vectors is still unknown.

To understand the model used, the problem will be better defined: ASR deals with an undefined number of feature vectors on every utterance, since each of them has variable length which is unknown beforehand. As a consequence, it is not straightforward to apply techniques such as support vector machines or neural networks due to this fact. Instead, a model which is able to represent and identify temporal changes is needed. Graphical models are suitable for this task since they are able to represent probabilities and logical structure in terms of graphs. In this framework, Hidden Markov Models (HMMs), are suitable to represent temporal sequences.

As a reminder, HMMs consist of states that have transition probabilities among them; after transitioning, a symbol is emitted. More formally, an HMM H is defined as

$$H = (Q, \Sigma, A, B, \pi) \tag{2.19}$$

where Q is the set of states of the model, Σ is the set of possible symbols that can be emitted, A is the transition matrix that holds probabilities of transitioning from a state q_i to another state q_j, B is the emission matrix which stores the probability of emitting any symbol o_i in a given state q_i, and π is a vector that establishes the initial state probability [34]. HMMs are based on the Markov assumptions:

1. The state where the model is at time t, q_t, only depends on the state where the model was at time $t-1$, q_{t-1}.

2. The probability of transitioning at time t from a state q_t to another state q_{t+1} is independent of the time t in which this decision is taken. That is, these probabilities are stationary.

3. The probability of emitting a symbol o_t at time t on a state q_t only depends on the state q_t itself. This probability is also independent of the previous symbols emitted $o_1^{t-1} = o_1 o_2 \ldots o_{t-1}$.

The previous definition and assumptions result in a first order HMM. These assumptions hugely restrict the model, but they greatly help when formalizing the definitions, as it will be seen in the following paragraphs.

HMMs oriented to ASR are usually composed of three states, emit acoustic features and are usually represented by means of a strictly linear topology, in which there are transitions to the own state (loops) and to the next one. In principle, they could be identified by phones[12], each tri-state HMM representing a different phone. However, additional context for the identification of a phone is desired, since the phone's context may modify its respective spectrogram, which is represented in a (slightly) different acoustic feature. This is the main reason why triphones are introduced. Triphones are phones that consider their context both to the left and to the right, and are commonly defined as p_1–p_2+p_3, where p_2 is the phone to be identified, and both p_1 and p_3 are the phone's context.

The introduction of triphones carries additional problems. If there are N phones in a certain language, there will be N^3 triphones. Some of them will never (or barely) be seen in the training phase. Therefore, there will be some HMM states of the different triphones that will not have enough information to estimate their own emission probabilities. This is why HMM states are grouped by their acoustic similarity with respect to the HMM states of other triphones, which gives rise to the tied state triphone, or senone [18].

Senones are used in order to identify the corresponding probability function of a certain HMM state which can be shared among different HMM states. They are usually grouped through a Classification And Regression Tree (CART), which normally contains linguistic information and can be queried to know the senone that a certain HMM state points to. Figure 2.6 shows a graphical representation of this type of parameter sharing with Gaussian mixture models as emission probability distribution functions, also called GMM-HMM model.

Keeping in mind the information given in the previous paragraphs, the HMM that models a word w can be obtained by finding the sequence of triphones that compose w, and then concatenating together the HMMs corresponding to that sequence. Similarly, one can build an HMM for a sentence w_1^m by concatenating the HMMs obtained by the different words that compose the sentence (a process that will be formally defined in Section 2.9). Since this HMM emits acoustic features, it is possible to obtain

[12]Technically, one could model phones: "distinct speech sound or gesture, regardless of whether the exact sound is critical to the meanings of words", as well as phonemes: "speech sound [...] that, if swapped with another phoneme, could change one word to another". As it can be seen, both definitions are quite similar, and only differ from a word perspective. However, in order to follow the literature, the "phones" notation will be used from now on. Source for the definition of phone and phoneme: `https://en.wikipedia.org/wiki/Phone_(phonetics)`. Last accessed: 11-09-2021.

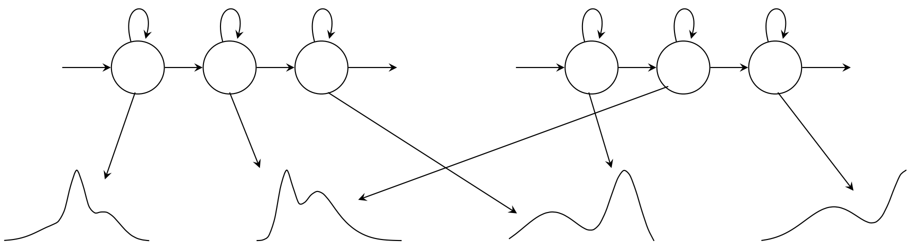


Figure 2.6: Graphical representation of parameter sharing by the HMM states. The emission probabilities of each HMM state are modeled by the senones, and each senone may model one or more HMM states' emission probabilities.

$P(a_1^n \mid w_1^m)$[13] by applying the Viterbi formula:

$$\begin{aligned} P(a_1^n \mid w_1^m) &= \sum_{s_1^n \in S} P(a_1^n, s_1^n) = \sum_{s_1^n \in S} P(s_1^n) P(a_1^n \mid s_1^n) \\ &\approx \sum_{s_1^n \in S} \prod_{i=1}^{n} P(s_i \mid s_{i-1}) P(a_i \mid s_i) \end{aligned} \tag{2.20}$$

where S is the set of every possible state sequence that may appear in the corresponding HMM (note that the length of each sequence must be exactly n). In the previous equation, $P(s_i \mid s_{i-1})$ denotes the transition probabilities (obtained by the matrix A in the HMM definition), while $P(a_i \mid s_i)$ references the emission probabilities (obtained by the matrix B in the HMM definition).

As it has been previously introduced, the emission probabilities of the HMMs can be estimated through Gaussian mixture models, which gives rise to the GMM-HMM model. However, more modern approaches use a DNN as a probability estimator. This yields some advantages, the most obvious of which is better results because of the DNN's strong representation power [57]. Another advantage worth mentioning is a more noise-robust model [43].

The combination of HMMs and DNNs creates the DNN-HMM hybrid model, also called CD-DNN-HMM (Context-Dependent DNN-HMM) hybrid model if senones are being modeled. In this case, because of the disciminative ability of the DNNs, the

[13] w_1^m is "syntactic sugar" that refers to the state sequence of the HMM built with the sentence w_1^m, so for simplicity it will be discarded after the first equality for better readability of the whole equation.

formula that supports this model slightly changes with respect to Equation 2.20[14]:

$$\begin{aligned}
P(a_1^n \mid w_1^m) &= \sum_{s_1^n \in S} P(a_1^n, s_1^n) = \sum_{s_1^n \in S} P(s_1^n) \cdot P(a_1^n \mid s_1^n) \\
&\approx \sum_{s_1^n \in S} \prod_{i=1}^{n} P(s_i \mid s_{i-1}) \cdot P(a_i \mid s_i) \\
&= \sum_{s_1^n \in S} \prod_{i=1}^{n} P(s_i \mid s_{i-1}) \cdot \frac{P(s_i \mid a_i) \cdot P(a_i)}{P(s_i)}
\end{aligned} \tag{2.21}$$

Again, S refers to the set of every possible state sequence of the HMM, and $P(s_i \mid s_{i-1})$ references the transition probabilities of the HMM. However, by applying the Bayes' formula on $P(a_i \mid s_i)$, more terms appear: $P(s_i \mid a_i)$ can be obtained by examining the output of the DNN whose input is a_i (which implies that the network acts as a senone classifier), $P(s_i)$ is the *a priori* probability of the senone, and $P(a_i)$ is the probability of finding the acoustic vector a_i. However, it is reminded that this probability is included in a maximization (Equation 2.8 or 2.9). $P(a_i)$ is the same regardless of the word sequence w_1^m, and therefore it can also be discarded.

As it has been introduced in the previous paragraph, the DNN which models the probability $P(s_i \mid a_i)$ acts as a senone classifier. This can be enforced by setting a softmax as the last layer's activation function, which guarantees a probability distribution among all outputs. Note that in this case, for the network to be able to classify senones, the number of neurons in the output layer should be equal to the number of senones.

To conclude this section, it is worth noting that an architecture based on HMMs is not the only possible architecture in order to address ASR. Other architectures have emerged over time, such as transducer models [6]. Transducers are end-to-end models, which means that they directly learn to map a sequence of acoustic features to a sequence of graphemes or words through a single system, instead of relying on acoustic and language models separately [50].

2.8 Acoustic modeling: DNN-HMM hybrid approach

As it was previously mentioned, as of today HMMs are generally paired with DNNs. However, it is common to use architectures more advanced than the feed-forward network, in which the output of a layer is the input of another one, and information can only go forward. Recurrent-based architectures, which give support to this work and will be analyzed in this section, have become the state of the art in acoustic modeling. However, there are other promising architectures which will be briefly discussed as well.

[14]The same reasoning as Footnote 13 has been applied.

2.8.1 Recurrent-based acoustic modeling

As previously mentioned, an RNN can be used as the acoustic model in the hybrid DNN-HMM model. This implies that both the LSTM and the BLSTM models can also work as the acoustic model. This work will focus on the so-called BLSTM-HMM hybrid model [15] by developing several acoustic models based on the BLSTM architecture. This has been the predominant hybrid model for a long time, achieving many successes throughout the years [20, 27].

As a note, a drawback of this model when used for real-time acoustic modeling is that, theoretically, the whole utterance should be obtained before starting the decoding process because of how the backward part of a BLSTM works. As a consequence, some models are proposed based on limiting the future information to N frames. An effective idea is the one proposed by the latency-controlled BLSTM, in which the backward part can only look N frames ahead [58].

2.8.2 Other acoustic model types

Attention-based models have recently gained traction in ASR. However, the state-of-the-art Transformer cannot be directly used in hybrid systems as the acoustic model, since the original Transformer is a sequence-to-sequence architecture based on attention. In [51], several layers inspired in the Transformer are stacked together, creating an acoustic model and substituting the BLSTM in the BLSTM-HMM hybrid architecture. According to the original paper, this yields improvements over the BLSTM-HMM architecture.

Convolutional Neural Networks (CNNs) have also been used in acoustic modeling. In the case of acoustic modeling, they require features which are correlated in time and frequency, which means that MFCC vectors are not suitable to be used because of the decorrelation obtained after the DCT step. However, using data in the spectral domain points to improvements with respect to the DNN-HMM architecture [3, 41].

Those readers interested in a detailed study of recent techniques for acoustic modeling are addressed to [58].

2.9 Decoding

The decoding or inference step is the process by which the system extracts a sequence of labels given a sequence of inputs. In the case of ASR, labels correspond to words, while the inputs correspond to the acoustic vectors extracted in the preprocessing stage.

In hybrid models, this can be done by creating a Finite State Transducer (FST). An FST is similar to a Finite State Automaton (FSA), but instead of having a single symbol on each edge which the automaton "follows", each edge now has an input symbol, an output symbol. Two types of FSTs particularly interesting in this case are the Weighted Finite State Transducer (WFST) and the Weighted Finite State Acceptor (WFSA). A WFST also has in each edge a weight which maps symbols from input to output (which acts as a transition cost). Therefore, a path in a WFST maps a sequence of input symbols to a sequence of output symbols [28]. A WFSA is a WFST whose input and output symbols in each edge are the same.

Let $\circ$ be the composition operator for FSA but applied to WFST and WFSA (described in [28]); H a WFST which maps HMM states to context-dependent phones; C a WFST which maps context-dependent phones to phones; L a WFST which represents a lexicon that maps phones to words; and G a WFSA which represents a grammar that obtains gramatically correct sentences with respect to the language desired to recognize. The composition

$$N = H \circ C \circ L \circ G \tag{2.22}$$

yields as a result an FST N which maps HMM states to sequences of words restricted to the grammar G [28]. However, a more compact FST is desired for a faster decoding. This can be obtained by optimizing the computation of N through a series of operations. Let $min(\cdot)$ and $det(\cdot)$ be the equivalent of the minimization and determinization operations for FSA but applied to WFST and WFSA (also described in [28]). The composition

$$N' = min(det(H \circ min(det(C \circ min(det(L \circ G)))))) \tag{2.23}$$

yields as a result an FST $N' \equiv N$ which is more compact than N, and therefore traversing it can be done in a more efficient way. Figure 2.7 shows the basic composition of these structures in the decoding step.

This FST can be represented by a trellis or decoding graph, depicted in Figure 2.8. In this trellis, the horizontal direction represents the different feature vectors extracted from the utterance during the preprocessing step, and the vertical direction represents all HMM states product of the previously shown composition. In the trellis represented by HMMs following the strictly linear topology, there are transitions from a state q_i to another state q_j in these conditions:

1. Whenever both states are part of a triphone HMM. These are the usual transitions inside the tri-state HMM.

2. Whenever the lexicon L allows it. This lexicon splits a word into its respective phones, from which triphones can be extracted. In this case, q_i must be the final state of the HMM that models a triphone x, q_j must be the beginning state of the HMM that models a triphone y, and x must be found before y in any of the words in the lexicon.

This trellis can be analyzed by means of the Viterbi algorithm, an algorithm which may be optimized by dynamic programming. The final word sequence uttered can be computed during the search of the best path by examining the lexicon, and finding the sequence of phones that form each word. Even then, it is very costly to keep in memory the whole search process due to the fact that, in the worst case, V potential words can start at each time t, being V the length of the considered vocabulary. Moreover, using non-trivial language models (such as trigrams) increase the search space [35].

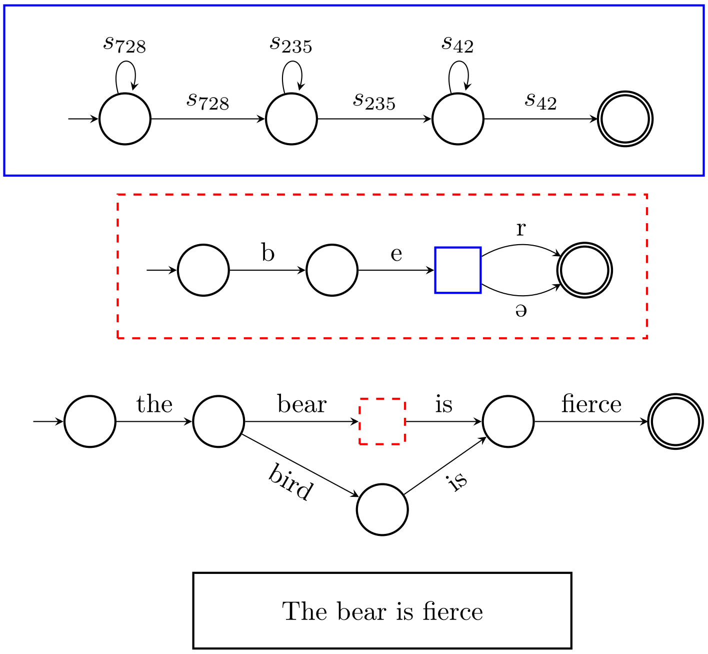


Figure 2.7: Composition of the different FSTs H, C, L and G. The weights and the output symbols have been omitted for clarity. Squares represent information coming from the previous level. The dashed square represents the two phonetic transcriptions of the word “bear”, /ber/ and /beə/, which have been obtained from the Cambridge dictionary (`https://dictionary.cambridge.org/`).

In practice, an algorithm known as beam search is combined with the Viterbi algorithm. This algorithm stores the N best paths and prunes the rest, being N a parameter of the algorithm called beam size. Although optimality is not guaranteed, in practice it allows an effective and efficient decoding, since it removes from the search all sequences which are less likely to succeed.

Still, the searching process is expensive with a complex language model. Nowadays, a two-step decoding process has become popular, in which the first pass is done with a relatively basic language model, and in the second pass the most promising paths are rescored with a bigger language model [29, 22]. However, a one-step decoding process with LSTM language models subject to real-time constraints has been achieved by using implementation tricks which are able to speed up the process, such as not computing all outputs of the softmax layer at each step [22]. This was later replicated with Transformer language models in [5].

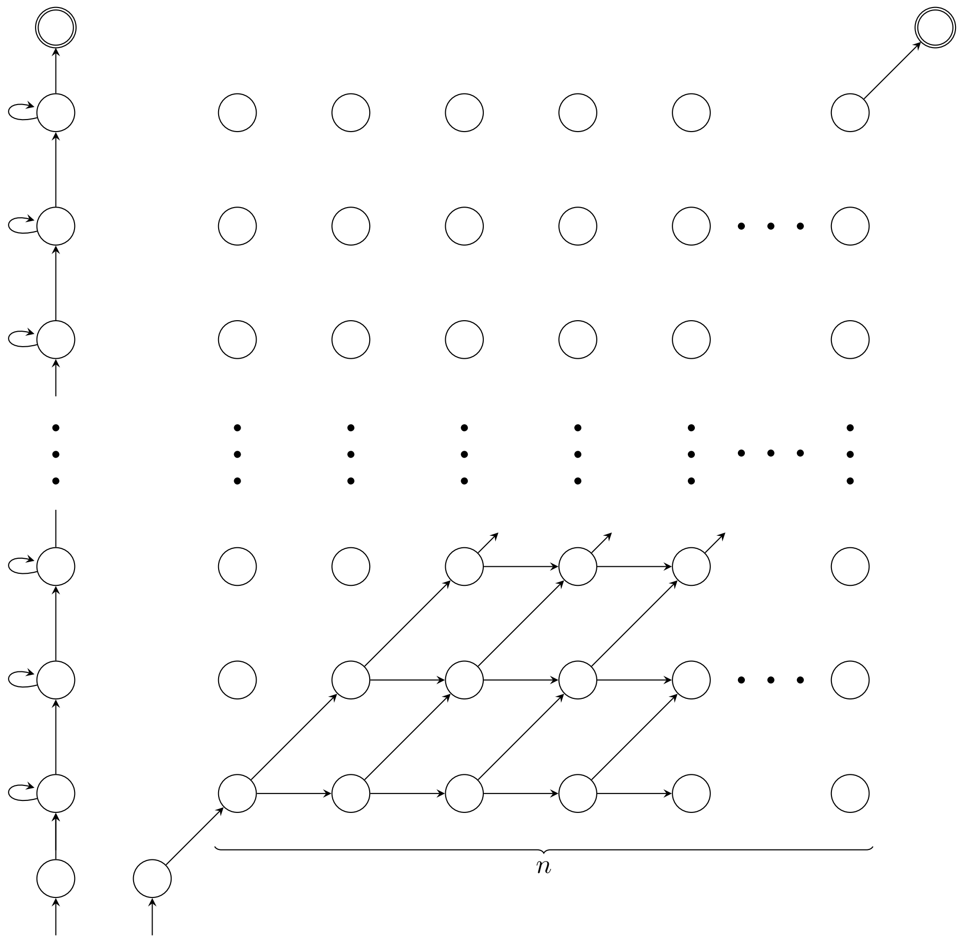


Figure 2.8: Representation of the decoding step by means of a trellis. The first column represents the concatenation of the different HMMs. This trellis is divided into n sections, being n the number of frames or acoustic features from the sample.

2.10 ASR Evaluation

In ASR, it is essential to have universal measurements that are able to compare different systems. The most common evaluation measurement is the Word Error Rate (WER), which compares the minimum amount of insertions, deletions and substitutions needed for the obtained transcription to become the reference transcription and is calculated in the following way:

$$\text{WER} = \frac{I + D + S}{|R|} \tag{2.24}$$

where $|R|$ is the length of the reference transcription measured in words, and I, D and S are the minimum number of insertions, deletions and substitutions needed for the system output to become the reference transcription R. This measurement is similar to the Levenshtein distance and it is usually multiplied by 100 in order to yield a percentage. Notice that it could reach a figure higher than 100% since the potential number of insertions I is unbounded.

Another evaluation measurement is the Senone Error Rate (SER), defined in the following way:

$$\text{SER} = \frac{|S_{\text{inc.}}|}{|S_{\text{tot.}}|} \tag{2.25}$$

where $|S_{\text{inc.}}|$ and $|S_{\text{tot.}}|$ are the number of incorrect and total senones analyzed respectively. This measurement is not as common as the WER, but it will be of use in this work in order to establish a comparison between the performance of the different acoustic systems during the training phase.

Chapter 3

Project description, dataset and tools used

This chapter describes the main projects in which this work has been developed. Moreover, the speech dataset in which this work has been supported is analyzed. Finally, the core tools used in this work are introduced.

3.1 Multisub and CAR research projects

In this section, both of the projects that support this Master's Thesis, **Multisub** and **Classroom Activity Recognition**, are described. Both projects are created in order to boost Open Education, while Multisub also addresses Parliamentary Openness.

Open Education (OE), also known as the Open Educational Resources (OER) movement, promotes a new style of teaching and learning through the use of new technologies. It aims at providing open access to high-quality educational content, usually digital, on a global scale. This can be achieved by the negligible cost of reproduction derived from the widely extended usage of technology [8].

Parliamentary Openness (PO) is defined as "increased commitment to openness and to citizen engagement in parliamentary work" [53]. It strives to promote a culture of openness, achieve transparency in parliamentary information, make this kind of data more accessible to the general public and enable its electronic communication [53].

As it will be seen in Section 3.2, due to the nature of the dataset, it is expected that the systems developed as a part of this work focus on OE. However, these systems could also be applied to PO.

3.1.1 Multisub

The first project which enables the work described in this document is **Multilingual subtitling of classrooms and plenary sessions** (in the following **Multisub**), a three-year[1] project funded by the Ministerio de Ciencia, Innovación y Universidades de España under reference RTI2018-094879-B-I00 (MCIU/AEI/FEDER, EU), addressed by the MLLP-UPV group.

The objective of this project is to generate high-quality automatic transcriptions and translations of real-life scenarios by using both ASR and MT technologies. In this sense, OE and PO are the core application areas of this project.

The main challenges of this work related with the **Multisub** project are the following:

- Channel variability noise and reverberation: as it will be detailed in 3.2, there are two sources of recordings (clip-on and camera), one of which is noisier than the other. However, it is desired that the different systems developed work nicely for both types of audio, and specially for the noisier source of audio.
- Far-field speech recognition: some microphones allow capturing long-range interactions, the source of which is located far away from the device. An example of this could be a student asking a question to the lecturer.
- Multitalker recognition: in an utterance, there may be more than one speaker, such as in a classroom with teacher-student interactions. It is desired that the resulting ASR system recognizes all of the interactions.

The project also covers other interesting points related to ASR:

- On-line speech recognition: an on-line ASR system should be able to transcribe sentences as they are being uttered, that is, it should work in a real-time environment, which implies additional constraints.
- Speaker diarization: it is the process that segments an utterance in different parts depending on who is speaking, which can be used for instance to classify a class recording on teacher or student interventions.

3.1.2 Classroom Activity Recognition

Another project in which this work has been carried out is the **Classroom Activity Recognition** (in the following **CAR**) project, a four-year[2] project funded by the Conselleria d'Educació, Investigació, Cultura i Esport de la Generalitat Valenciana within the Prometeo grant for research groups of excellence under reference PROMETEO/2019/111, which was addressed by the Valencian Research Institute for Artificial Intelligence (VRAIN).

The main objective of the **CAR** project which can be found in this work is the provision of high-quality automatic transcriptions of classroom video recordings. As in the **Multisub** project, this also implies far-field speech recognition, as well as speech enhancement by trying to find a mapping between noisy and clean audio.

Other relevant objectives considered by the project are:

[1]Grant period: 01/01/2019 - 31/12/2021.
[2]Grant period: 01/01/2019 - 01/01/2023.

- Provision of an automatic classification of activities from classroom transcriptions: an automatic classifier of different types of class interactions needs to be built. Readers interested in this topic are addressed to [37], which highlights recent work made in this direction.
- Provision of a behavioral pattern model of students and lecturers: this objective explores causal and temporal dependencies among the different class interactions detected, and intends to create behavior models for teachers and students.
- Provision of an academic performance assessment model: the models are analyzed, and discrete variables such as "time spent on explaining theory", "number of posted questions by the lecturer" or "number of asked questions by female/male students" can then be extracted and subsequently examined in order to determine the most influential factors.

3.2 Videoapuntes dataset description

Videoapuntes is a project launched by the UPV in 2012 with the main aim of producing high-quality classroom recordings which include audio, video and the slides used by the teacher. The resulting recordings are made available online on the virtual UPV platform PoliformaT. The video repository in which these recordings are stored, also called Videoapuntes, contains as of today more than 50000 hours of video recordings, a number which greatly increased these last years due to the COVID-19 pandemic and the fact that most of the classes were recorded.

The dataset used in this work is a representative subset of Videoapuntes composed of a total of 487 classes lectured at the UPV, which is translated to more than 700 hours of recordings in total. However, audio is duplicated since there are two main sources of audio which record the same class at the same time: a lapel or clip-on microphone ("clip-on microphone" or simply "clip-on" from now on), and a microphone integrated in the webcam available in the class ("camera microphone" or simply "camera" from now on). This means that the real length of the dataset is double the previous amount, roughly 1400 hours.

Even though both the clip-on and the camera microphones record the same lecture, they do so differently. The clip-on microphone only captures the lecturer's voice and features both integrated noise reduction and high audio quality. In contrast, the camera microphone captures a lot of background noise but it is intended to be seen as a far-field microphone, that is, a microphone which is able to capture sounds whose origin is located far away, like interventions from students.

In this regard, it is interesting to cite the paper that explored data augmentation by artificially injecting noise: "[...] learning multiple types of noises [in DNNs] are not only possible, but also complementary" [55]. A DNN, thanks to its discriminative capabilities, is able to learn complex boundaries even in the presence of noise and recognize them separately. This gives support to the theory which states that the acoustic model developed in this work could be trained with both sources of audio, performing great in both of them at the same time.

In real-life situations, the camera microphone is the most common source of audio due to the fact that it is very common for a classroom of the UPV to be equipped with a camera which is able to stream the lecture; the camera is usually in charge of transmitting both audio and video. However, it is less common for the clip-on microphone to take an active part in a lecture, since it must be requested and sometimes synchronized beforehand. This is why it is desirable that the developed systems perform nicely in the camera part of the dataset, even when it is noisier than the clip-on counterpart.

In the dataset there is a large volume of unlabeled data available (train partition or set), and there is also a smaller volume of manually transcribed data (development[3] and test partitions or sets). The transcriptions of the development and test partitions were manually obtained by workers of the Servipoli Foundation[4]. It was not part of this work to split the data into partitions, which were already given when the work described in this document started. Table 3.1 shows the duration of the whole dataset split in the respective train, dev and test partitions.

Table 3.1: Initial duration of the dataset, split in partitions.

Partition	# lectures	Clip-on rec. hrs.	Cam. rec. hrs.	Total hrs.
Train	475	740.5	740.2	1480.7
Dev	6	11.7	11.7	23.4
Test	6	11.2	11.2	22.4

However, having two separate sources of real-life data carries additional problems: data coming from both sources may not be perfectly synchronized, there are recordings where the audio coming from the clip-on microphone is cut, there are audios which only contain background noise, or some recordings end abruptly, among other problems. The most conservative approach, which has been taken in this work, is to discard those recordings that show any anomaly similar to the ones stated above.

Discarding a recording took place if the offset between the clip-on and the camera recording was too long. This offset was calculated by minimizing the difference of energy and the zero-crossing rate[5] between the clip-on and the camera recordings. This reduced the available number of videos in the train partition from 475 to 433. Table 3.2 shows the remaining duration of the dataset. The reader should notice how the duration of both the clip-on and the camera recordings now reflect the same figure in the train set, as opposed to Table 3.1.

As previously mentioned, most of the data was unlabeled, which is why the training transcriptions were obtained from the unlabeled train set by applying the most recently developed Spanish ASR system by the MLLP-UPV research group, trained with nearly 4000 hours of data coming from different sources [21]. This system will

[3]In most of the document, this partition will be denoted as "dev".

[4]`www.servipoli.es`. Last accessed: 20-06-2021.

[5]"Rate at which a signal changes from positive to zero to negative or from negative to zero to positive". Source: `https://en.wikipedia.org/wiki/Zero-crossing_rate`. Last accessed: 11-09-2021.

Table 3.2: Duration of the dataset after synchronizing clip-on and camera audio, split in partitions.

Partition	# lectures	Clip-on rec. hrs.	Cam. rec. hrs.	Total hrs.
Train	433	655.1	655.1	1310.2
Dev	6	11.7	11.7	23.4
Test	6	11.2	11.2	22.4

be analyzed in Section 4.2.

The MLLP-UPV system recognized the clip-on recordings, which allowed the obtention of acceptable base transcriptions on both sources of data: the clip-on transcriptions were used as the camera transcriptions since both sources were previously synchronized. If the camera recordings had been transcribed with the baseline system, the resulting transcriptions would have been much worse. This allowed the problem to become a supervised one with recordings and their respective transcriptions, even if these transcriptions were unsupervised. Obtaining these transcriptions was not done by the author, and it is outside of the scope of the document.

3.3 Tools used

This section details the main tools used in order to develop the final acoustic models: the transLectures-UPV toolkit (TLK) and TensorFlow.

3.3.1 TLK

TLK [10] is an ASR toolkit capable of performing every task in the ASR pipeline: acoustic preprocessing, training, and decoding. It is able to preprocess audio samples, train GMM-HMM and DNN-HMM hybrid acoustic models, including BLSTM-HMM models, and allows the integration of language models trained externally to the decoding step.

As a tool, TLK offers three high-level scripts which are defined in the following:

- **tLtask-preprocess**: it is in charge of preprocessing the samples in order to extract the respective feature vectors. It can yield MFCC vectors, as well as filter bank vectors.

- **tLtask-train**: it takes the feature vectors previously generated and trains a model. It is able to train monophone and triphone models, as well as to convert the triphone models into tied phone models and to perform speaker adaptation. Internally, it uses the Baum-Welch and the Viterbi algorithms for parameter estimation.

- **tLtask-recognise**: it is able to obtain the most probable hypothesis for a sequence of acoustic vectors. To do so, it performs the Viterbi algorithm as explained in Section 2.9.

3.3.2 TensorFlow

TensorFlow [2] is an open-source library created for machine learning. It enables an easy environment of creation and training of machine learning models. When developing a model, TensorFlow uses dataflow graphs in order to represent both the computation in a specific algorithm and its state.

In this work, TensorFlow was used to train BLSTM-HMM acoustic models, since training a neural network in TLK is internally performed by TensorFlow. Moreover, the labels (senones corresponding to each acoustic vector) were converted into TFRecords, a TensorFlow format for storing sequential binary records[6] which the network can later process.

[6]Source: `https://www.tensorflow.org/tutorials/load_data/tfrecord`. Last accessed: 11-09-2021.

Chapter 4

Acoustic model training

This chapter details the process done in order to obtain the different ASR systems which will be evaluated in the next chapter. First, an introduction to the training pipeline and the strategies followed to train the different acoustic models will be provided. Then, the training pipeline will be explained, and after that, the baseline ASR system will be evaluated in order to check the performance of the acoustic system with respect to the speech dataset. Finally, the training phase of the different acoustic models will be described.

In this chapter and the following ones, some special terms will be used for clarity:

- The terms "clip-on audio" and "camera audio" will be used in order to refer to audios captured by the clip-on microphone or the camera microphone, respectively.
- The terms "clip-on WER" and "camera WER" will denote the WERs obtained in the clip-on and the camera recordings, respectively.

4.1 Introduction

As it was seen when describing the speech dataset, there are two sources of audio: clip-on and camera. Given this fact, several training strategies can be followed to obtain competitive systems. In this work, two training strategies were followed:

1. The first one was to train from scratch different types of acoustic models in order to compare their performances with respect to the acoustic model of the baseline ASR system. In this regard, two types of acoustic models were trained:
    - In order to check the behavior of the ASR system with respect to isolated clip-on audios, the first one was trained from scratch only with clip-on audio.
    - The second acoustic model was trained with both clip-on and camera audio. This system is justified by the fact that it is desired to evaluate the impact

of using both sources of audio data with respect to the usage of isolated clip-on audios.

2. The second strategy applied was to fine-tune the acoustic model of the baseline ASR system with both clip-on and camera audio. The fine-tuning process will be thoroughly described in Section 4.6, but in short, it consists in using a previously trained model in order to adapt it to the training set.

Moreover, in the context of the **Multisub** project, two types of acoustic models were also developed (trained and fine-tuned) taking into account only camera recordings. However, they were not part of the work described in this document since they were not developed by the author.

4.2 Baseline system

Table 4.1 shows the resources used in order to train the baseline acoustic model. This table was obtained from [21, Table 1].

Table 4.1: Resources used to train the acoustic model by the MLLP-UPV research group.

Resource	Duration (h)
Internal: entertainment	2932
Internal: educational	406
Internal: user-generated content	202
Internal: parliamentary data	158
RTVE2018 contest datasets	205
VoxForge[1]	21
Total	3924

The first thing to be done was to establish the baseline of system performance to be compared with the rest of the experiments. This baseline was obtained with the latest MLLP-UPV Spanish ASR system[2], consisting of a BLSTM-HMM hybrid system with eight BLSTM layers. Table 4.2 shows the WERs achieved with this system with respect to the dev and test partitions.

Table 4.2: WERs obtained by the baseline Spanish ASR system in the dev and test sets.

	Clip-on WER (%)	Camera WER (%)
Dev	11.6	57.4
Test	12.1	60.4

[1]VoxForge: Free Speech... Recognition. `http://www.voxforge.org`. Last accessed: 11-09-2021.
[2]As a reminder, this system was also used to extract the training transcriptions.

These are already satisfactory results on the clip-on recordings. The camera recordings are hard to be recognized due to their high background noise and considerably lower quality than the clip-on recordings, which explains their higher WERs. The development of the diverse systems, covered in the following sections, will focus on lowering these error rates, mostly on the camera setting.

4.3 Training pipeline

This section outlines the sequence of steps taken in order to train the different acoustic models.

Figure 4.1 shows a summary of the training pipeline followed in order to train the different systems described. Assuming that the training transcriptions exist, the usual acoustic modeling pipeline consists of the following steps: first, since the network is trained discriminatively, the transcriptions must be aligned in order to have samples with its respective labels; then, the network training (in this case the BLSTM training) can be performed.

However, in this case the transcriptions of the train partition were not available. To circumvent this problem, it was decided to obtain them in an unsupervised way, using the latest MLLP-UPV Spanish ASR system in order to produce these transcriptions in a reasonable time with acceptable quality.

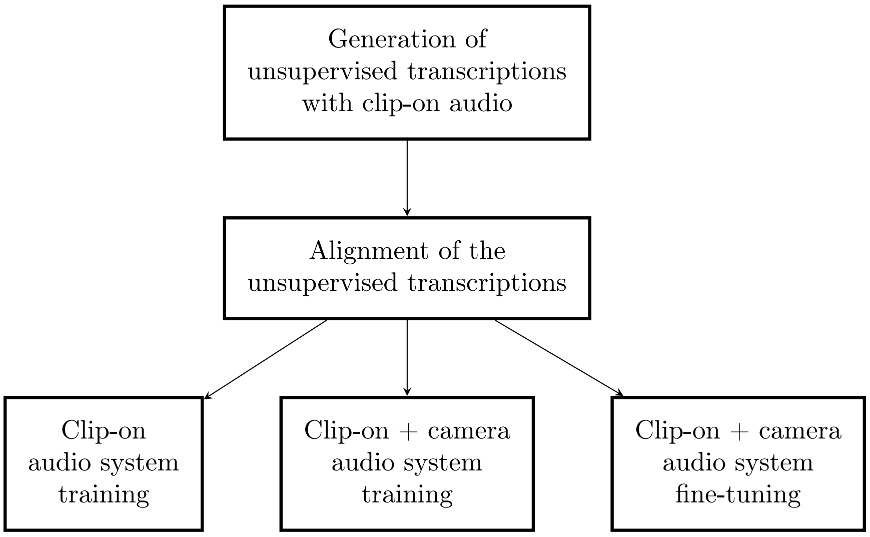

Generation of unsupervised transcriptions with clip-on audio
Alignment of the unsupervised transcriptions
Clip-on audio system training
Clip-on + camera audio system training
Clip-on + camera audio system fine-tuning

Figure 4.1: Summary of the sequence of steps taken in order to train the different systems.

The following subsections will analyze in detail the first two steps. The proper training phase of the different acoustic models will be covered in different sections.

4.3.1 Generation of unsupervised transcriptions with clip-on audio

As it was mentioned on Chapter 3, most of the classroom video recordings do not contain transcriptions. This part corresponds to the train set, since the transcriptions of both the dev and the test sets were manually produced. However, it is very challenging to do the same in the train set, because it consists of more than 1300 hours of data. Manually transcribing this set demands a lot of resources and time, and it is practically not feasible unless a large-scale work schema is used such as crowdsourcing, which in turn is usually very expensive.

In order to obtain the training transcriptions, the baseline MLLP-UPV Spanish ASR system was used at the very beginning of the pipeline in order to obtain the training transcriptions. Then, these transcriptions were obtained in an unsupervised way, which makes the rest of the experiments a more challenging task.

Due to the low audio quality in the camera part, the transcriptions were obtained with respect to the clip-on part of the dataset. These transcriptions could also be used in the camera part because data coming from both parts was assumed to be the same.

4.3.2 Alignment of the unsupervised transcriptions

Before starting the proper training phase of the systems, the transcriptions must be aligned. This process, usually called forced alignment, obtains the most probable label (senone or tied phoneme) for each acoustic vector. The model used to align the transcriptions was the baseline BLSTM model. Notice that only the senones corresponding to the acoustic vectors are needed, which is why an HMM is not needed to do this work. The BLSTM aligns the data by using the corresponding acoustic vector as input and choosing the most probable senone obtained in the output layer.

This process used the clip-on acoustic vectors because this audio had a lower WER than its camera counterpart, which is an indication that it is understood in a more reliably way by the model. If the camera recordings had been used for this, the senones obtained in this alignment would have been of a poorer quality than the ones obtained with the clip-on recordings.

The alignment could be used for the camera part by simply replacing the acoustic vectors of the clip-on recordings by the ones of the camera recordings, in the exact same order. This could be achieved because, when cleaning the dataset, the clip-on and the camera recordings were forced to be perfectly synchronized as well, as explained in Section 3.2.

4.4 Features of the developed acoustic models

The first acoustic model developed in this work is composed of four BLSTM layers of 1024 neurons for each layer: 512 analyze the acoustic features in the forward sense and 512 analyze it backwards. The number of BLSTM layers was an element analyzed throughout the experiments, as it will be seen later on. Between the last BLSTM layer and the output layer, there is a bottleneck layer of 200 neurons.

The output of the model is the probability of every senone appearing in the acoustic vector passed as input. As a consequence, in the output layer there are 10041 neurons, which corresponds to the number of senones. The activation function of the output layer is a softmax activation function, in order to yield a probability distribution at the end of the network. The hidden layers have no explicit activation function because the activation function is already embedded in the LSTM cell, as previously seen in Figure 2.2.

The system is trained with a batch size of 40. This means that 40 samples are processed before changing the internal parameters of the network. When the training has gone through every sample of the dataset, it is said that an epoch has passed. The training algorithm iterates through several epochs until convergence or until manually stopped.

Moreover, the technique known as dropout was also applied, which disables (drops) a certain percentage of neurons in the training phase so that the rest of the neurons learn to generalize, preventing overfitting [45]. In this work, the dropout rate has been set to 0.1, meaning that 90% of the neurons were active during the training phase at each time step.

Since a classification problem is being tackled, the cross-entropy loss function has been applied during the training phase:

$$CE(w) = -\frac{1}{N}\sum_{n=1}^{N}\sum_{c=1}^{C} t_{n,c} \cdot \log s_c(x_n; w) \tag{4.1}$$

where w is the weight vector to be found by optimizing the function, N is the number of samples, C is the number of classes (in this case, the number of senones), x_n is the n-th sample of the dataset, $t_{n,c}$ is the c-th dimension of the one-hot encoded vector[3] refering to the n-th sample, and $s_c(x_n; w)$ is the c-th output of the last layer of the neural network, which is $P(c \mid x_n)$ assuming a softmax activation.

In all of the trainings, the system used the Adam optimizer [24], starting with an initial learning rate α = 5e-5. This learning rate will be modified in a specific experiment later on. Moreover, the system has a learning rate decay $d = 0.5$. The learning rate decay is applied whenever the results obtained in the current epoch are not improved with respect to the previous ones, being the new learning rate on the next epoch $\alpha_{t+1} = \alpha_t \cdot d$.

As for data augmentation, SpecAugment was used by masking 27 frequency channels twice, and again masking 10 time steps twice, which makes a total of four maskings. In the decoding part, beam search has been applied with a beam size $N = 150$.

[3]This vector has C dimensions; it has a one in the respective class dimension, and a zero in the rest of the positions. Therefore, every class is encoded in its corresponding dimension.

4.5 First training strategy: training from scratch

This section describes the process done to train the systems from scratch. Both types of systems will be mentioned, showing their respective performances in the training phase.

4.5.1 Clip-on audio system

The first system explicitly trained as a part of this work used as training data only the clean audio coming from the clip-on microphone. As a reminder, these are more than 650 hours of clean audio with no background noise.

The system was trained for a total of 15 epochs. Figure 4.2 shows the error obtained on the development partition with respect to the training epoch.

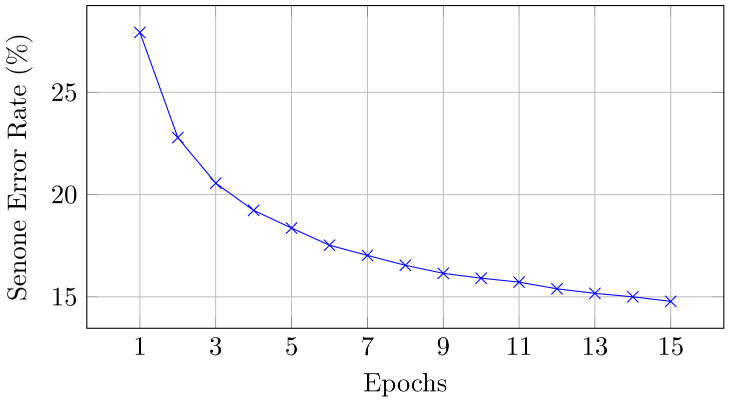


Figure 4.2: SER obtained per epoch in the development partition when training the clip-on audio system from scratch.

4.5.2 Clip-on and camera audio system

This system used as training data not only the clean audio coming from the clip-on microphone, but also the noisy audio coming from the camera microphone, which means that it used the whole train partition in the training phase. Ideally, it would be able to learn audio features from the clean audio while learning the background noise pattern present in the camera recordings, being able to perform nicely in both types of audios.

Three different configurations of this system were trained with 4, 6 and 8 BLSTM layers respectively. All of the systems were trained for a total of 15 epochs in order to compare them to the previous results. As a side note, considering models with more layers increases the time taken for the model to process samples, which implies a higher train and recognition time. Figure 4.3 shows the error obtained in the development partition with respect to the training epoch.

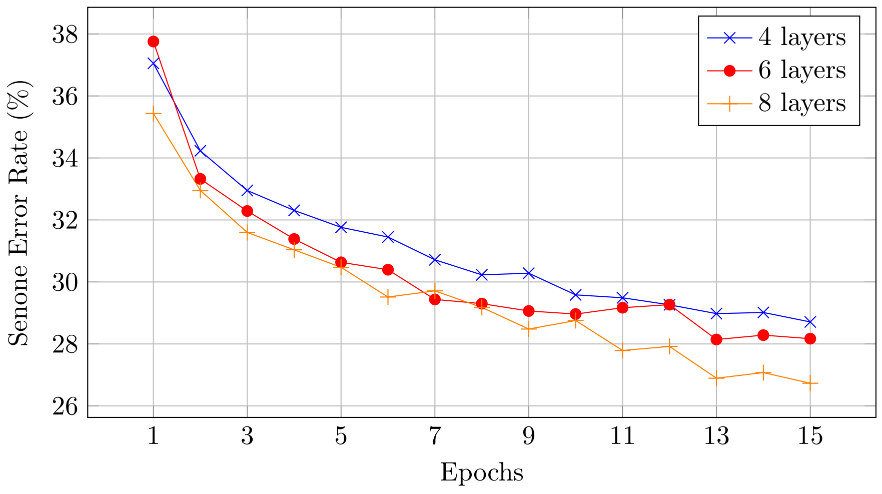


Figure 4.3: SER obtained per epoch in the development partition when training the clip-on and camera systems from scratch, with respect to the number of layers.

The error rates in the dev partition still converge smoothly, but they reach a minimum error rate which is higher than the one shown in Figure 4.2. This is expected due to the fact that, as it was already established, the camera recordings show worse acoustic features than the clip-on recordings. The error rate in Figure 4.3 could thus be taken as a sort of average between the error rates already seen when training the clip-on recording system and the error rates which would be seen when training the camera recording system.

4.6 Second training strategy: fine-tuning of the baseline MLLP-UPV Spanish ASR system

Fine-tuning an acoustic model means taking an acoustic model which was already trained, usually with data from other domains unrelated to the task, and then resume training considering the partitions of the dataset as usual. This is done in order to obtain a model whose parameters are better tuned to the task. The former model is referred to as the *pretrained model*, while the latter model is called the *fine-tuned model*.

Fine-tuning a system falls inside the set of techniques known as *transfer learning*. Transfer learning takes features learned on a specific problem, and uses them in order to tackle another similar problem. In this schema, the pretrained model can have more layers added and the previous ones can be frozen (usually the ones closer to the input so that they do not change their way of representing the samples), being the fine-tuning step an optional last step [9].

Theoretically, the fine-tuning of a system should be done with a learning rate lower than the one used to train the system, since it is desired to slightly readjust the weights of the network. However, because of the uncertainty of these fine-tuning experiments[4] and because the learning rate was already low enough, two different systems were trained: one with a learning rate α = 5e-5 (the same as the previous systems), and another with a learning rate one magnitude lower: α = 5e-6.

The pretrained model used was the baseline MLLP-UPV Spanish acoustic model, whose description was already given in Section 4.2. The fact that this model had 8 BLSTM layers increased the time taken in order to finish an epoch. Even then, this model was already trained for 16 epochs, so the resulting models were expected to perform better than the previously developed models.

Both models were fine-tuned using the whole training set (clip-on and camera speech data) for 15 epochs. Figure 4.4 shows the error rate in the training phase of both fine-tuned systems.

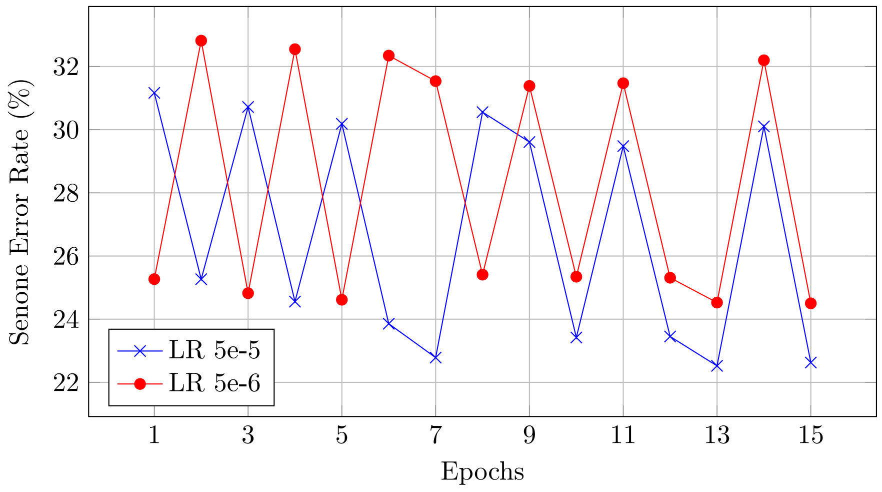


Figure 4.4: SER obtained in the development partition when fine-tuning the systems with an initial learning rate α = 5e-5 and α = 5e-6.

As it can be seen, the first system (trained with a learning rate α = 5e-5) oscillates in terms of error rate and it does not smoothly converge, which is usually caused by a high learning rate. However, this behavior still applies to the second system (of a learning rate α = 5e-6). The non-convergence could be caused by the fine-tuning of the systems with respect to the camera recordings, which are a difficult type of audio to treat, knowing that the baseline system was not trained in noisy conditions.

However, even if the systems did not converge, the results obtained were not poor.

[4]The MLLP-UPV group had not applied fine-tuning to its systems before, so the behavior of the model with respect to the learning rate when fine-tuning was unknown. This experiment was also meant as a test of whether a lower learning rate was really needed for them to fine-tune an acoustic model.

In fact, the minimum error rate of the first system in the dev set during training is 22.5% (13th epoch), while the best error rate achieved by the second system is 24.5% (15th epoch), two absolute points higher. A lower error rate may help achieve a smoother convergence, but it could also force higher error rates in the training phase, as seen from the previous data.

Chapter 5

ASR system evaluation

For the acoustic models to be evaluated in terms of WER, they have to be integrated with a language model, creating an ASR system. This chapter details the process carried out to evaluate each of the ASR systems with respect to the dev and the test partitions. First, the steps taken in order to evaluate these systems will be explained. Then, some of the parameters to be optimized will be introduced. After that, the different acoustic models developed will be integrated into an ASR system and evaluated in terms of WER. Finally, the best results obtained will be discussed.

5.1 Introduction

A high validation error in the training phase by the acoustic models may hint at a lower performance, but the resulting ASR systems need to be evaluated in order to check their performance. In order to evaluate the WERs of the different systems, the acoustic model has to be integrated with a language model. Therefore, the evaluation is not based only on the acoustic model anymore, but on the ASR system as a whole.

Figure 5.1 shows a summary of the steps taken in order to evaluate the different developed systems. Before the evaluation, a set of training experiments have to be performed in order to optimize different decoding and model parameters. The criterion to optimize these parameters is the WER obtained in the dev partition with respect to the set of parameters that need to be optimized. When these optimal parameters are obtained, an unbiased evaluation with respect to the test partition is needed in order to check if the parameters obtained in the dev partition are still valid with respect to a totally different partition which has not been explicitly optimized.

The language model used in the decoding step was already given as part of the baseline ASR system and it was not part of this work to obtain it, but its details are shown here. It is an interpolation of a Transformer and n-gram[1] language models, with weights 93% and 7% respectively. The n-gram was chosen to be of order 4

[1]An n-gram language model takes a window of $n-1$ words in order to compute the probability of the word w_n as $P(w_n \mid w_1^{n-1})$. The parameter n is also called the n-gram order.

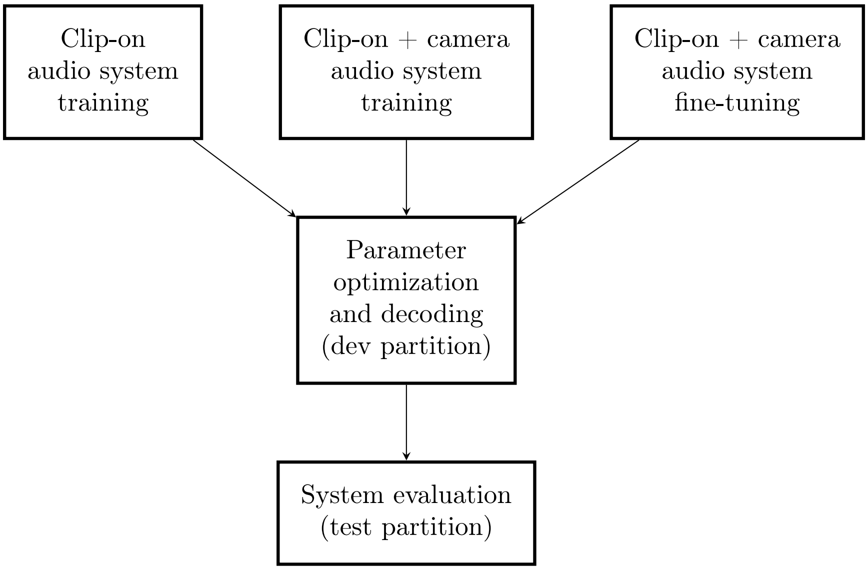

Clip-on audio system training
Clip-on + camera audio system training
Clip-on + camera audio system fine-tuning
Parameter optimization and decoding (dev partition)
System evaluation (test partition)

Figure 5.1: Summary of the sequence of steps taken in order to evaluate the different systems.

and it was trained with 102G running words coming from different sources, while the Transformer was trained with a subset of roughly 1G running words. Those interested in the extraction process or in more details are pointed to [21, 5].

5.2 Parameter optimization

Two of the most well-known parameters to be optimized after training the systems are the Grammar Scale Factor (GSF)[2], and the Word Insertion Penalty (WIP):

- The GSF scales the probability of the language model, since depending on the task, the language model can hold more or less importance. If this parameter was not present, the acoustic model could dominate over the language model.
- The WIP is added as a constant that multiplies the number of words extracted by the system in order to reward or punish the system by extracting too many or too few words.

These are added to the optimization formula of ASR from Equation 2.9, yielding the following equation:

$$\hat{w} = \operatorname*{argmax}_{w_1^m \in W^*} \log P(a_1^n \mid w_1^m) + \beta \cdot \log P(w_1^m) + \gamma \cdot m \tag{5.1}$$

[2]Also found in the literature as Language Model Scaling (LMS).

where β is the GSF and γ is the WIP. Currently, both parameters are learned experimentally by testing different values. However, in the experiments it was decided to only modify the GSF, since it yields more variability in terms of error rate. Therefore, in all of the experiments, the WIP parameter was considered equal to zero.

Moreover, one can also experiment with the senone priors. As it was established when explaining the fundamentals of acoustic systems in Section 2.7, the priors establish the *a priori* probability for each senone. This probability is used in order to regularize senones: since the term is dividing, the most frequent senones are penalized with respect to the least frequent senones, which are rewarded. The following section will detail the process carried out in order to extract the priors for each senone.

5.3 Senone prior extraction

As it was introduced in the previous section, the priors hold the *a priori* probability of each senone. In TLK, priors are specified in the form of a text file, each line having a log-probability for each of the senones. This log-probability can be easily related to each senone, since the latter are identified by a number $0 \leq i < N$, where N is the maximum number of senones.

Four types of priors have been extracted in order to experiment with them:

1. No priors at all. This is the case where the denominator $P(s_i)$ in Equation 2.21 is ignored. This was done because it is desirable to see the practical effect of the priors in the recognition.

2. Priors coming from the baseline ASR system. Since they are already computed, they were also considered in the experiments.

3. Priors coming from the alignments of the training partition. They are supposed to be a reliable way of computing the priors, since they represent the dataset's senone distribution. They were easily obtained by calculating the probability of each senone in the labels used to train the different systems.

4. Priors from the alignments, with the log-probability of the silence (sil) lowered. In the original priors coming from the alignments, $\log P(sil) = -0.578723$. Since the probability of the silence was overestimated with respect to the rest of the senones (with an average log-probability near -11) due to the dataset's features, it is desired to tone this probability down. The log-probability of the silence was subtracted two, four and six points, yielding three different sets of priors which were also tested in the experiments.

5.4 Evaluation of the clip-on ASR system trained from scratch

The first experiment consisted in modifying the GSF parameter and the priors file, aiming to optimize the WER obtained in the clip-on part of the dataset. Figures 5.2

and 5.3 show the results obtained with respect to the development partition by considering different GSF parameters and the prior sets described in Section 5.3. Each point in the graph corresponds to a full recognition of the corresponding part of the dev set. The six types of priors described in Section 5.3 were tested. Moreover, a minimum choice of three GSF parameters (8, 10, 12) was tested, decreasing this parameter by two until reaching a clear error curve in every execution. This made a total of 26 recognitions of the dev set (both the clip-on and the camera parts).

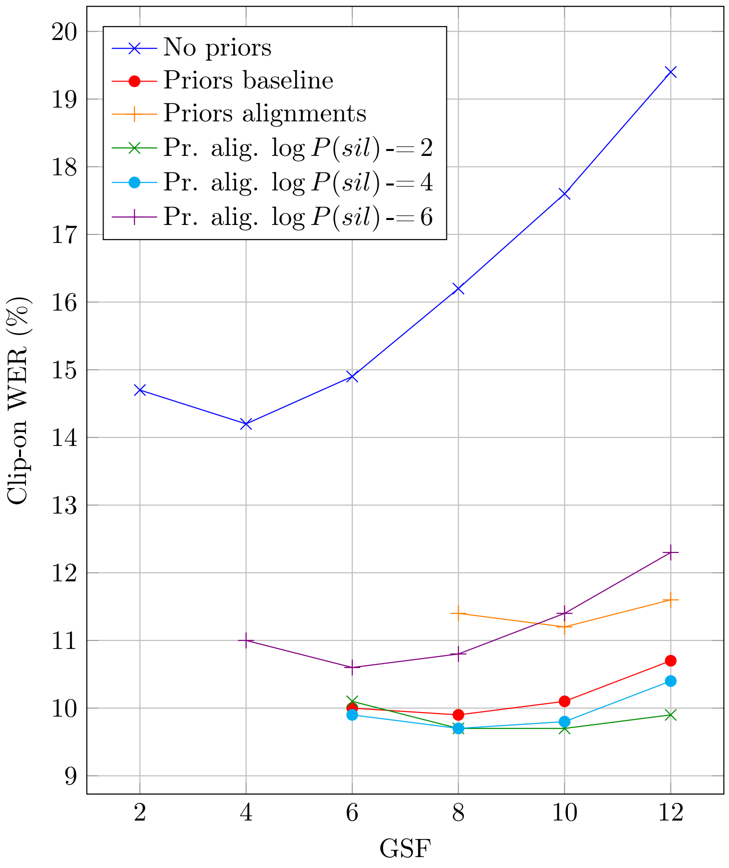


Figure 5.2: WERs obtained in the clip-on part of the dev set with respect to the GSF parameter and the priors considered.

As it can be seen, modifying the GSF impacts the WER in a smooth way in the clip-on part of the dataset by establishing a very clear error curve. However, this situation is different in the camera part, decreasing the WER even at a GSF = 2 and behaving erratically in some executions. This is because the camera part has not been considered during the training phase, so this type of audios is unknown to the system.

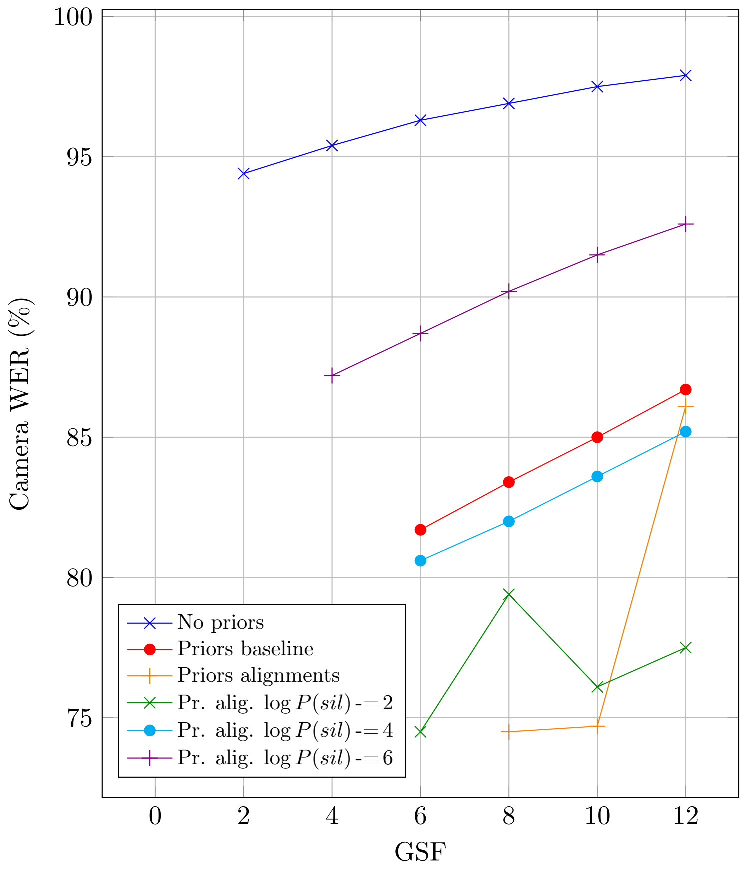


Figure 5.3: WERs obtained in the camera part of the dev set with respect to the GSF parameter and the priors considered.

Choosing the right set of priors proved to be relevant from these results: neglecting the priors resulted in losing 4.5 WER points with respect to the best clip-on audio system. Moreover, tuning special priors such as the prior of the silence resulted in a positive change up to a point: giving it too much relevance, as well as too little relevance, led to a decrease of the performance of the system. This could have happened because the silence plays a major role in the recorded lectures since a lecturer often pauses between phrases, while there are other situations which may have also been taken as a silence by the clip-on microphone such as a student asking or answering a question.

The best WERs were obtained with a GSF = 8, with the priors coming from the alignments where the silence log-probability was subtracted two or four points. With these results, the WERs of the test partition were obtained with respect to the best system: the one using GSF = 8 and the priors coming from the alignments where the log-probability of the silence was subtracted two points (the tie was solved by choosing the priors which had a better performance in the camera partition). Table 5.1 shows these results.

Table 5.1: WERs obtained in the test set with respect to the best parameters obtained in the optimization phase.

Clip-on WER (%)	Camera WER (%)
10.3	76.9

After this first experimentation, the decrease in WER obtained with respect to the baseline was of 1.8 points (14.9% relative decrease) in the clip-on recordings of the test set. However, the camera WER failed to improve with respect to the baseline, with an increase of 16.5 points (27.3% relative increase). The next experiments will mostly focus on lowering the camera error rates since, as previously stated, the camera system is the most common system found in classroom video recordings.

5.5 Evaluation of the clip-on and camera audio system trained from scratch

Figures 5.4 and 5.5 show the WERs obtained with respect to the priors which obtained the best results in both the camera and the clip-on recordings when experimenting with the clip-on audio system: the priors coming from the alignments of the training set where the log-probability of the silence was subtracted two points. This was done to avoid a highly dense experimentation. Still, in this case, four recognitions of the whole dev set were needed for each of the three systems, making a total of 12 recognitions of the dev set.

Comparing the results given in these figures against the ones previously given in Figures 5.2 and 5.3, it can be seen that the clip-on WER slightly varies with respect to the clip-on audio system: it increases 0.3 points in the system with four BLSTM layers, and it decreases 0.4 points in the system with eight BLSTM layers. However, the camera WER achieves an astonishing decrease with respect to the former system developed, decreasing 45.7 points in the system with four BLSTM layers and 47.3 points in the system with eight BLSTM layers.

Finally, Table 5.2 shows the WERs obtained in the test set with respect to the layers of the system and GSF = 8. The chosen value of the GSF parameter was the one that yielded the best results in the camera part of the experiment.

In the test partition, the relative results with respect to the clip-on audio system are still consistent with those obtained in the dev partition. In the system with four BLSTM layers, the clip-on WER increases by 0.1 points but the camera WER significantly decreases by 44.2 points. This behavior is similar to the system with eight BLSTM layers, where the clip-on WER decreases by 0.3 points and the camera WER decreases by 48.6 points. By including the camera recordings to the training

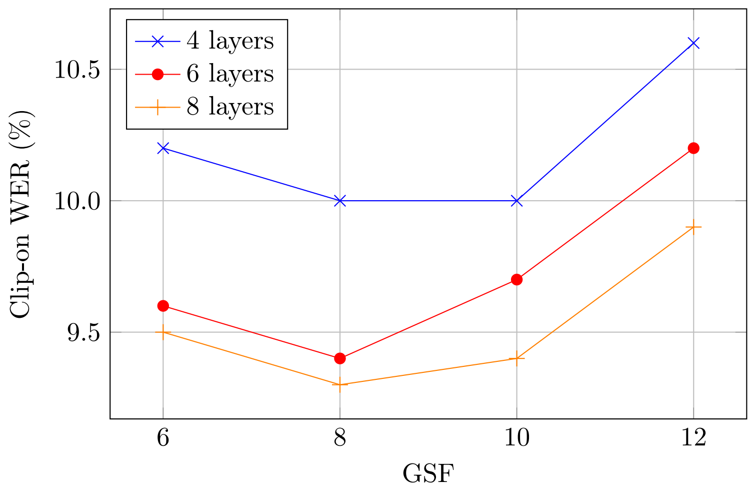


Figure 5.4: Different WERs obtained in the clip-on part of the dev set with the systems trained with the clip-on and the camera recordings, with respect to the GSF parameter.

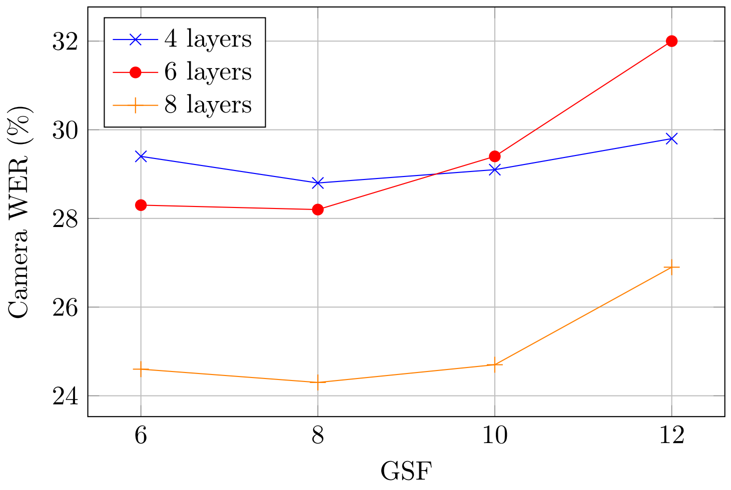


Figure 5.5: Different WERs obtained in the camera part of the dev partition with the systems trained with the clip-on and the camera recordings, with respect to the GSF parameter.

data of the model, the system effectively learns to better distinguish the relevant acoustic features even in the presence of background noise.

With respect to the baseline, the relative results are outstanding: the system with four BLSTM layers achieves a decrease of 1.7 points of WER (14.0% relative decrase)

Table 5.2: WERs obtained in the test set with respect to the best parameters obtained in the optimization phase.

# layers	Clip-on WER (%)	Camera WER (%)
4	10.4	32.7
6	10.2	31.4
8	**10.0**	**28.3**

in the clip-on part of the test set, and of 27.7 points (45.9% relative decrease) in the camera part. In the system with eight BLSTM layers this situation is even more positive, with a decrease in terms of WER of 2.1 points (17.4% relative decrease) in the clip-on part, and of 32.1 points (53.1% relative decrease) in the camera part.

5.6 Evaluation of the fine-tuned ASR system

Figures 5.6 and 5.7 show the behavior of both systems in terms of WER. Again, the priors which performed the best in the first experiment were used: the priors obtained from the alignments, with the silence log-probability subtracted two absolute points. In this case, the clip-on part of the dev set was recognized a total of nine times, while the camera part was recognized eight times.

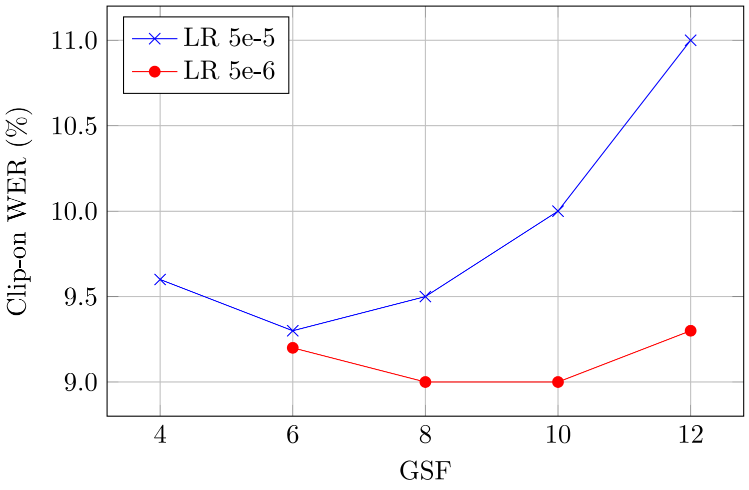


Figure 5.6: WERs obtained in the clip-on part of the dev set with the fine-tuned ASR systems, with respect to the GSF parameter.

In this case, the system that performed the best in the clip-on recordings in terms of WER was the second system (of learning rate α = 5e-6), obtaining a difference of 0.5 points with respect to the first system (of learning rate α = 5e-5). However, the

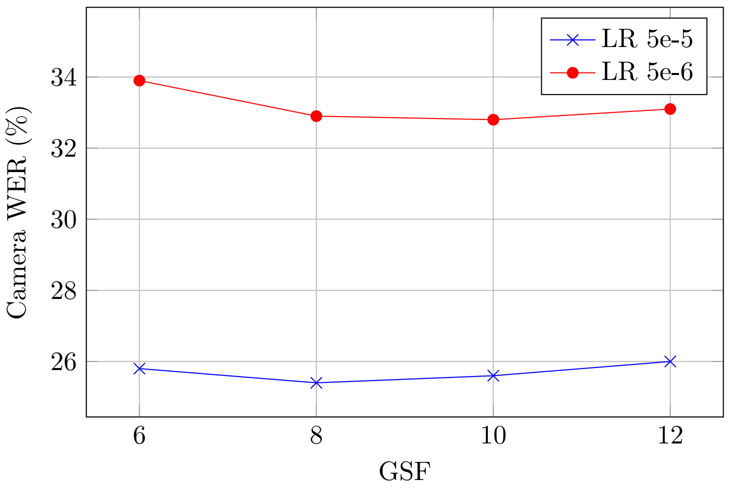


Figure 5.7: WERs obtained in the camera part of the dev partition with the fine-tuned ASR systems, with respect to the GSF parameter.

first system reached much better WERs in the camera recordings than the second one, with a difference of 7.4 points.

Finally, Table 5.3 shows the WERs extracted by both systems with respect to the test partition. Since the main aim of this experiment was to optimize the results in the camera recordings, the GSFs used to recognize the test partition were the ones which yielded the best result in the camera part of the experiment for each system: GSF = 8 for the system with learning rate α = 5e-5, and GSF = 10 for the system with learning rate α = 5e-6.

Table 5.3: WERs obtained by the fine-tuned ASR systems in the test set with respect to the best parameters obtained in the optimization phase.

Initial LR	Clip-on WER (%)	Camera WER (%)
5e-5	**9.7**	**29.2**
5e-6	10.0	37.2

In the case of the test set, the system of learning rate α = 5e-5 performs much better than the system of learning rate α = 5e-6. This could be caused by the fact that the weights of the network were altered more easily with a higher learning rate, which benefited the test set the most since it was not seen during the training phase.

Moreover, the increased WER obtained in the camera recordings by the system of learning rate α = 5e-6 could be explained by the fact that the baseline system was trained with cleaner data than the classes recorded by the camera: even if both systems did not smoothly converge in the training phase, the system of learning rate α = 5e-5 could have been able to adapt better to the camera recordings thanks to its higher learning rate, which is shown in the 8 absolute WER points of difference in the test set.

The relative results with respect to the baseline system are great, but they do not improve the previous best results in the camera setting: the fine-tuned system of learning rate α = 5e-5 achieves a decrease in WER of 2.4 points (19.8% relative decrease) in the clip-on part of the test set, and of 31.2 points (51.7% relative decrease) in the camera part. The fine-tuned system of learning rate α = 5e-6 achieves a decrease in WER of 2.1 points (17.4% relative decrease) in the clip-on part, and of 23.2 points (38.4% relative decrease) in the camera part.

5.7 Discussion

Table 5.4 summarizes the best WERs obtained for all the ASR systems developed with respect to the test set. Note that the results obtained with the systems developed only with camera recordings are also displayed. Even if these were not extracted in the work done by the author, it is useful to compare the different results.

Table 5.4: Summary of the best WERs obtained in the test set and relative decrease of the WER from the baseline by each of the developed systems in the context of the **Multisub** and the **CAR** projects. The systems in italics correspond to systems not developed by the author. The executions where the baseline was not improved are displayed as "0.0".

Strategy	ASR system	# layers	WER (%)		% decrease wrt. baseline	
			Clip-on	Camera	Clip-on	Camera
–	Baseline	8	12.1	60.4	–	–
From scratch	Clip-on	4	10.3	76.9	14.9	0.0
	Camera	6	100.2	29.7	0.0	50.8
	Clip-on + camera	4	10.4	32.7	14.0	45.9
		6	10.2	31.4	15.7	48.0
		8	10.0	**28.3**	17.4	**53.1**
Fine-tuning camera	*α = 5e-5*	8	100.2	29.7	0.0	50.8
	α = 5e-6		99.2	33.7	0.0	44.2
FT[3] clip-on + cam.	α = 5e-5	8	**9.7**	29.2	**19.8**	51.7
	α = 5e-6		10.0	37.2	17.4	38.4

Overall, the best performing system was the system trained from scratch with eight BLSTM layers, closely followed by the fine-tuned system with a learning rate α = 5e-5. The former system achieved the best WERs out of all systems in the camera part of the dataset, while the latter system obtained the best WERs in the clip-on part.

[3]Fine-tuning.

Adding the camera recordings to the training set helped the overall WER, as shown in the changes from the clip-on audio system to the clip-on and camera audio systems. Moreover, the only cases where the system was not able to improve the baseline were those systems that only had information about one audio setting (either clip-on or camera). This may point to the fact that, even if audio sources are quite different, it is benefitial to consider each of the sources when training, provided that the recognition of such sources is essential.

The systems trained only with camera recordings have the worst clip-on WERs, but they show an excellent performance in the camera recordings, reaching less than 30% WER in two of the three developed systems. Like the system trained only with clip-on recordings, they failed to relate acoustic features to the opposite type of recordings.

With respect to the camera recordings, it was surprising to find that the fine-tuned clip-on and camera system performed (slightly) worse than the systems trained from scratch with camera audios, given that the fine-tuned system was pretrained with roughly 4000 hours. As hinted before, this may be due to the fact that the corpora that were used when training the pretrained acoustic model (already shown in Table 4.1) were clean enough, and they could not be linked to the camera part of the dataset by the acoustic model.

This leads to the hypothesis that a fine-tuned system may not only need a suitable learning rate or more epochs in the fine-tuning phase, but also to have seen samples similar to the ones to be recognized, especially in terms of external conditions such as background noise. In this regard, one could train a system with data augmentation based on adding artificial noise to the samples. However, this was not the case in the baseline system, because it was (and is) intended to be used in controlled, almost noise-free scenarios.

Chapter 6

Conclusions

In this work, the transcription of classroom recordings has been explored through different techniques that have proven useful to accomplish the main goals. The major problems found during the course of this work have been described and solved. Two main types of systems have been developed from scratch, each addressing different audio sources: one which was trained only with the clip-on recordings and through which an initial exploration was performed, and several others which were trained with both the clip-on and the camera recordings. Moreover, the baseline system was fine-tuned with respect to both types of recordings. In all of the systems, an exploration of the best system parameters was performed, and the different WERs were reported. Finally, the best results from each system were gathered and discussed.

Both sources of audio were mixed in the training of state-of-the-art BLSTM-HMM acoustic models, which allowed the different ASR systems to obtain a good performance in both types of recordings. These systems obtain high-quality transcriptions in the case of the clip-on recordings and acceptable transcriptions in the case of the camera recordings, improving the baseline ASR system in most of the cases. Furthermore, they could be used in order to transcribe classroom video recordings, as seen from the WERs of the test partition. Therefore, it can be said that all proposed objectives were fulfilled.

By looking at the error rates of the three types of developed systems in this work, it can be said that it was benefitial for the system to include the noisier samples, even more when these were an essential part of the recognition: the camera WER dropped from a 76.9% in the case of the clip-on audio system to a 21.6% in the clip-on and camera ASR system with the same number of BLSTM layers, while the clip-on WER stayed near a 10%. This can be an indication that data coming from different sources can be mixed in the training process so that the acoustic model learns to identify both at the same time, which may acquire relevance when there is more than one type of audio to be recognized by the model, as it was the case in this work.

In terms of WER, it is benefitial to use the system trained from scratch with eight BLSTM layers for the camera recordings and the fine-tuned system with learning rate $\alpha = 5\text{e-}5$ for the clip-on recordings, but it depends on the situation whether or not to use it. Using a system with eight BLSTM layers may be a disadvantage if recognition speed or memory consumption are a key factor to be taken into account. If that was the case, one could then use systems with four BLSTM layers instead of eight, thus sacrificing some performance (0.6% WER on clip-on recordings and 4.4% on camera recordings, choosing the clip-on and the clip-on + camera ASR systems trained from scratch, respectively) but obtaining an overall less consuming decoding. This could be the case, for instance, when using the system in a real-time environment knowing in advance that the lecturer will use a clip-on microphone, or in environments with limited resources.

Moreover, fine-tuning a system may help recognize samples provided that they are of a similar nature, for instance in the image domain. Fine-tuning may not be the best choice if there is a considerable distance between the samples used to pretrain the system and the samples used to fine-tune the system, but it may be very useful if there is not a huge amount of data available. Data augmentation based on adding background noise to the recordings used in the pretraining phase could also be applied, yielding a very noise-robust pretrained system which could then be fine-tuned.

6.1 Future Work

Future work could focus on performing experiments with isolated recordings with different background noise conditions, in order to check if the systems trained with the noisier sources of audio have properly learned to separate the actual utterance from the background noise (and if so, to which extent), or rather the systems simply "memorized" the background pattern of the camera recordings.

Another experiment could also involve adding more layers to the system trained from scratch with both the clip-on and the camera samples, and see the point where the systems stop learning because of the depth of the network. This experiment could be enhanced by testing other architectures for the acoustic system, such as CNNs, a Transformer acoustic encoder, or other architectures.

Moreover, the baseline system is not perfect, which means that errors coming from the transcriptions of the unlabeled training data obtained by this system cannot be ignored. Future work could focus on minimizing these errors by, for instance, starting from the already established transcriptions and iteratively training systems (or fine-tuning either the baseline system or any of the systems extracted in this work) and recognizing again the training partition.

Finally, the language model was not modified. It would have been desirable to adapt the language model to the academic and scientific domain, which best fits the available data. A possibility would be retraining from scratch the language model with data coming from different sources, as well as from the training transcriptions. Another possibility would be taking an already created language model and fine-tuning it, in a similar way to the acoustic model fine-tuning. A final, more difficult option would involve having several language models and choosing among them at the beginning of the lecture the one which fits best, or using dynamic language modeling techniques such as a neural cache-based language model [14].

Acknowledgements

The author would like to formally thank the projects **Multilingual subtitling of classrooms and plenary sessions** (**Multisub**), reference RTI2018-094879-B-I00 (MCIU/AEI/FEDER, EU), and **Classroom Activity Recognition** (**CAR**), reference PROMETEO/2019/111 (Conselleria d'Educació, Investigació, Cultura i Esport de la Generalitat Valenciana), in which this work has been developed.

BIBLIOGRAPHY

[1] New MLLP project: "Multisub: Multilingual subtitling of classrooms and plenary sessions", funded by the Government of Spain. `https://www.mllp.upv.es/new-mllp-project-multisub-multilingual-subtitling-of-classrooms-and-plenary-sessions-funded-by-the-government-of-spain/`, May 2019. Last accessed: 11-09-2021.

[2] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Łukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[3] Ossama Abdel-Hamid, Abdel rahman Mohamed, Hui Jiang, and Gerald Penn. Applying Convolutional Neural Networks concepts to hybrid NN-HMM model for speech recognition. *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4277–4280, 2012.

[4] Rosana Ardila, Megan Branson, Kelly Davis, Michael Henretty, Michael Kohler, Josh Meyer, Reuben Morais, Lindsay Saunders, Francis M. Tyers, and Gregor Weber. Common Voice: A massively-multilingual speech corpus. *CoRR*, abs/1912.06670, 2019.

[5] Pau Baquero-Arnal, Javier Jorge, Adrià Giménez, Joan Albert Silvestre-Cerdà, Javier Iranzo-Sánchez, Albert Sanchis, Jorge Civera, and Alfons Juan. Improved Hybrid Streaming ASR with Transformer Language Models. In *Proc. Interspeech 2020*, pages 2127–2131, 2020.

[6] Eric Battenberg, Jitong Chen, Rewon Child, Adam Coates, Yashesh Gaur, Yi Li, Hairong Liu, Sanjeev Satheesh, David Seetapun, Anuroop Sriram, and Zhenyao Zhu. Exploring neural transducers for end-to-end speech recognition. *CoRR*, abs/1707.07413, 2017.

[7] Christopher M. Bishop. *Pattern Recognition and Machine Learning.* Springer-Verlag, Berlin, Heidelberg, 2006.

[8] Tom Caswell, Shelley Henson, Marion Jensen, and David Wiley. Open Content and Open Educational Resources: Enabling universal education. *The International Review of Research in Open and Distributed Learning*, 9(1), Feb. 2008.

[9] François Chollet. Keras documentation: Transfer learning & fine-tuning. `https://keras.io/guides/transfer_learning/`. Last accessed: 11-09-2021.

[10] M. A. del Agua, A. Giménez, N. Serrano, J. Andrés-Ferrer, J. Civera, A. Sanchis, and A. Juan. The transLectures-UPV toolkit. In *Proc. of VIII Jornadas en Tecnología del Habla and IV Iberian SLTech Workshop (IberSpeech 2014)*, Las Palmas de Gran Canaria (Spain), 2014.

[11] European Telecommunications Standards Institute (ETSI). Speech Processing, Transmission and Quality Aspects (STQ); Distributed Speech Recognition; Extended front-end feature extraction algorithm; Compression algorithms; Back-end speech reconstruction algorithm, November 2003.

[12] Gonçal V. Garcés Díaz-Munío, Joan Albert Silvestre-Cerdà, Javier Jorge, Adrià Giménez, Javier Iranzo-Sánchez, Pau Baquero-Arnal, Nahuel Roselló, Alejandro Pérez-González de Martos, Jorge Civera, Albert Sanchis, and Alfons Juan. Europarl-ASR: A large corpus of parliamentary debates for streaming asr benchmarking and speech data filtering/verbatimization. In *Proc. Interspeech 2021*, Brno (Czech Republic), 2021.

[13] J.J. Godfrey, E.C. Holliman, and J. McDaniel. Switchboard: telephone speech corpus for research and development. In *[Proceedings] ICASSP-92: 1992 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 517–520 vol.1, 1992.

[14] Edouard Grave, Armand Joulin, and Nicolas Usunier. Improving neural language models with a continuous cache. *CoRR*, abs/1612.04426, 2016.

[15] Alex Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. Hybrid speech recognition with Deep Bidirectional LSTM. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 273–278, 2013.

[16] Gerhard Heinzel, Albrecht Rüdiger, and R Schilling. Spectrum and spectral density estimation by the Discrete Fourier transform (DFT), including a comprehensive list of window functions and some new flat-top windows. *Max Plank Inst*, 12, January 2002.

[17] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[18] Mei-Yuh Hwang and Xuedong Huang. Subphonetic modeling for speech recognition. In *Speech and Natural Language: Proceedings of a Workshop Held at Harriman, New York, February 23-26, 1992*, 1992.

[19] jim schwoebel. Voice datasets: A comprehensive list of open source voice and music datasets. https://github.com/jim-schwoebel/voice_datasets, 2013. Last accessed: 11-09-2021.

[20] Javier Jorge, Adrià Giménez, Pau Baquero-Arnal, Javier Iranzo-Sánchez, Alejandro Pérez-González de Martos, Gonçal V. Garcés Díaz-Munío, Joan Albert Silvestre-Cerdà, Jorge Civera, Albert Sanchis, and Alfons Juan. MLLP-VRAIN Spanish ASR systems for the Albayzin-RTVE 2020 speech-to-text challenge. In *Proc. of IberSPEECH 2021*, pages 118–122, Valladolid (Spain), 2021.

[21] Javier Jorge, Adrià Giménez, Pau Baquero-Arnal, Javier Iranzo-Sánchez, Alejandro Pérez, Gonçal V. Garcés Díaz-Munío, Joan Albert Silvestre-Cerdà, Jorge Civera, Albert Sanchis, and Alfons Juan. MLLP-VRAIN Spanish ASR Systems for the Albayzin-RTVE 2020 Speech-To-Text Challenge. In *Proc. IberSPEECH 2021*, pages 118–122, 2021.

[22] Javier Jorge, Adrià Giménez, Javier Iranzo-Sánchez, Jorge Civera, Albert Sanchis, and Alfons Juan. Real-Time One-Pass Decoder for Speech Recognition Using LSTM Language Models. In *Proc. Interspeech 2019*, pages 3820–3824, 2019.

[23] Daniel Jurafsky and James H. Martin. *Speech and Language Processing (2nd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2009.

[24] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

[25] Tom Ko, Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur. Audio augmentation for speech recognition. In *INTERSPEECH*, 2015.

[26] Richard P. Lippmann. Speech recognition by machines and humans. *Speech Communication*, 22(1):1–15, 1997.

[27] Christoph Lüscher, Eugen Beck, Kazuki Irie, Markus Kitza, Wilfried Michel, Albert Zeyer, Ralf Schlüter, and Hermann Ney. RWTH ASR systems for librispeech: Hybrid vs attention. *CoRR*, abs/1905.03072, 2019.

[28] Mehryar Mohri, Fernando Pereira, and Michael Riley. Weighted finite-state transducers in speech recognition. *Computer Speech & Language*, 16(1):69–88, 2002.

[29] Miroslav Novak. Evolution of the ASR decoder design. In Petr Sojka, Aleš Horák, Ivan Kopeček, and Karel Pala, editors, *Text, Speech and Dialogue*, pages 10–17, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[30] Chigozie Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall. Activation functions: Comparison of trends in practice and research for deep learning. *CoRR*, abs/1811.03378, 2018.

[31] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: An ASR corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210, 2015.

[32] Daniel S. Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D. Cubuk, and Quoc V. Le. SpecAugment: A simple data augmentation method for automatic speech recognition. *Interspeech 2019*, Sep 2019.

[33] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. The Kaldi speech recognition toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society, December 2011. IEEE Catalog No.: CFP11SRW-USB.

[34] L. Rabiner and B. Juang. An introduction to hidden markov models. *IEEE ASSP Magazine*, 3(1):4–16, 1986.

[35] Steve Renals. Search and decoding. `https://www.inf.ed.ac.uk/teaching/courses/asr/2011-12/asr-search-nup4.pdf`. ASR lecture 10. University of Edimburgh. Last accessed: 11-09-2021.

[36] Tony Robinson, M. Hochberg, and S. Renals. IPA: improved phone modelling with recurrent neural networks. *Proceedings of ICASSP '94. IEEE International Conference on Acoustics, Speech and Signal Processing*, i:I/37–I/40 vol.1, 1994.

[37] Alberto Romero Fernández. Teaching activity recognition in lectures delivered at UPV, 2021. Degree Final Work.

[38] Anthony Rousseau, Paul Deléglise, and Yannick Estève. TED-LIUM: an automatic speech recognition dedicated corpus. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 125–129, Istanbul, Turkey, May 2012. European Language Resources Association (ELRA).

[39] D. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[40] Roozbeh Sadeghian, J. David Schaffer, and Stephen A. Zahorian. Towards an automatic speech-based diagnostic test for Alzheimer's disease. *Frontiers in Computer Science*, 3:13, 2021.

[41] Tara N. Sainath, Abdel-rahman Mohamed, Brian Kingsbury, and Bhuvana Ramabhadran. Deep convolutional neural networks for LVCSR. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8614–8618, 2013.

[42] Mike Schuster and Kuldip K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[43] Michael L. Seltzer, Dong Yu, and Yongqiang Wang. An investigation of deep neural networks for noise robust speech recognition. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 7398–7402, 2013.

[44] Seyed Reza Shahamiri. Speech vision: An end-to-end deep learning-based dysarthric automatic speech recognition system. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 29:852–861, 2021.

[45] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[46] Andreas Stolcke and Jasha Droppo. Comparing human and machine errors in conversational speech transcription. *CoRR*, abs/1708.08615, 2017.

[47] Zoltán Tüske, George Saon, and Brian Kingsbury. On the limit of English conversational speech recognition. *CoRR*, abs/2105.00982, 2021.

[48] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc.

[49] Changhan Wang, Morgane Rivière, Ann Lee, Anne Wu, Chaitanya Talnikar, Daniel Haziza, Mary Williamson, Juan Miguel Pino, and Emmanuel Dupoux. VoxPopuli: A large-scale multilingual speech corpus for representation learning, semi-supervised learning and interpretation. *CoRR*, abs/2101.00390, 2021.

[50] Song Wang and Guanyu Li. Overview of end-to-end speech recognition. *Journal of Physics: Conference Series*, 1187:052068, 04 2019.

[51] Yongqiang Wang, Abdelrahman Mohamed, Duc Le, Chunxi Liu, Alex Xiao, Jay Mahadeokar, Hongzhao Huang, Andros Tjandra, Xiaohui Zhang, Frank Zhang, Christian Fuegen, G. Zweig, and M. Seltzer. Transformer-based acoustic modeling for hybrid speech recognition. *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6874–6878, 2020.

[52] Jan Wohlfahrt-Laymann, Hermie Hermens, Claudia Villalonga, Miriam M. R. Vollenbroek-Hutten, and Oresti Banos. Mobilecognitracker - A mobile experience sampling tool for tracking cognitive behaviour. *J. Ambient Intell. Humaniz. Comput.*, 10(6):2143–2160, 2019.

[53] World e-Parliament Conference 2012. Declaration on Parliamentary Openness. `https://www.openingparliament.org/declaration/`. Last accessed: 11-09-2021.

[54] Dominika Woszczyk, Stavros Petridis, and David Millard. Domain adversarial neural networks for dysarthric speech recognition. *CoRR*, abs/2010.03623, 2020.

[55] Shi Yin, Chao Liu, Zhiyong Zhang, Yiye Lin, Dong Wang, Javier Tejedor, Fang Zheng, and Yinguo Li. Noisy training for deep neural networks in speech recognition. *EURASIP Journal on Audio, Speech, and Music Processing*, 2015, 12 2015.

[56] Steve J. Young, D. Kershaw, J. Odell, D. Ollason, V. Valtchev, and P. Woodland. *The HTK Book Version 3.4*. Cambridge University Press, 2006.

[57] Dong Yu and Li Deng. *Automatic Speech Recognition: A Deep Learning Approach*. Springer Publishing Company, Incorporated, 2014.

[58] Dong Yu and Jinyu Li. Recent progresses in deep learning based acoustic models (updated), 2018.

[59] Luke Zhou, Kathleen C. Fraser, and Frank Rudzicz. Speech recognition in Alzheimer's disease and in its assessment. In *Interspeech 2016*, pages 1948–1952, 2016.

List of Figures

2.1 General structure of a neural network 8
2.2 Internal view of an LSTM memory cell 9
2.3 Unidirectional versus bidirectional connections 10
2.4 Structure of a basic BLSTM network oriented to speech recognition . 11
2.5 16-filter mel filter bank 15
2.6 Parameter sharing by the acoustic model's HMM states 18
2.7 Composition in the decoding step 22
2.8 Trellis representing the decoding step 23

4.1 Summary of the system training steps 33
4.2 Clip-on audio system: error figures obtained during training 36
4.3 Clip-on and camera audio systems: error figures obtained during training 37
4.4 Fine-tuned systems: error figures obtained during training 38

5.1 Summary of the system evaluation steps 42
5.2 Clip-on audio system: WER obtained in the clip-on part of the dev set 44
5.3 Clip-on audio system: error obtained in the camera part of the dev set 45
5.4 Clip-on and camera audio systems: WERs obtained in the clip-on part of the dev set 47
5.5 Clip-on and camera audio systems: WERs obtained in the camera part of the dev set 47
5.6 Fine-tuned ASR systems: error obtained in the clip-on part of the dev set 48
5.7 Fine-tuned ASR systems: error obtained in the camera part of the dev partition 49

List of Tables

3.1 Initial duration of the dataset, split in partitions 28
3.2 Duration of the dataset after synchronizing clip-on and camera audio, split in partitions 29

4.1 Resources used to train the baseline acoustic model 32
4.2 WERs obtained by the baseline Spanish ASR system in the dev and test sets 32

5.1 Clip-on audio system: WERs obtained in the test set 46
5.2 Clip-on and camera audio system: WERs obtained in the test set . . . 48
5.3 Fine-tuned ASR systems: WERs obtained in the test set 49
5.4 Summary of the best WERs obtained by each ASR system in the test set 50