Deep Learning for Automatic Speech Recognition
Project Report

Dargier Antoine
CentraleSupélec, MDS - SDI
antoine.dargier@student-cs.fr

Ponchon Martin
CentraleSupélec, MDS - SDI
martin.ponchon@student-cs.fr

February 4, 2023

Abstract

Automatic speech recognition has known significant progress in the past few years. The objective of this work is to present speech to text systems, and study how deep learning has enabled improvements in the field.

For the implementation of some of these algorithms, we chose to focus on the task of transcribing audio in French, for which resources are much more limited than for English. Our dataset includes recordings of a single people reading sentences from the novels Les Misérables by Victor Hugo and Arsène Lupin contre Herlock Sholmes by Maurice Leblanc.

After a literature review of the traditional speech recognition pipeline which is composed of several independent blocks, we studied end-to-end systems. We first implemented a Convolutional Neural Network with features based on Mel-frequency cepstral coefficients (MFCC), which gave inconclusive results. Secondly, we studied Connectionist Temporal Classification (CTC) which addresses the alignment problem, one of the main issues in speech recognition.

We concluded our work with a review of a more recent model based on self-supervised learning, wav2vec 2.0, which produces state-of-the-art results. On our dataset, and we achieved a WER of 21%, largely due to punctuation not always being correctly transcribed.

1 Introduction/Motivation

1.1 Interest

Automatic Speech Recognition (ASR), also known as speech to text, consists in converting spoken language as an audio signal into written text. While easy for humans, this task has historically been hard for machines.

ASR is a very dynamic field of research because of its utility in our daily life. Speech to text enables better human to machine interaction, and has numerous applications.

1.2 History

Research into speech recognition began in the mid-twentieth century. The first system to enter this field appeared in 1952 and was able to recognise the 10 digits. After that, research improved greatly with IBM and the work of Jelinek in the 1970s. In 1972 the first word recognition system was marketed by the company Threshold Technologies.

This science experienced a new boom in the 2000s, when Google added a voice-activated search tool to its search engine in 2008 and Apple added Siri to its phones from 2011. In 2017 Microsoft announced that it had achieved human voice recognition performance.

1.3 Fields of research

The topic is significant because we are then able to obtain a lot of information about the speech from the text: the content, the topic, the interlocutors, the tone, ... We can for instance use Natural Language Processing (NLP) algorithms on the texts to understand their content and context. So speech recognition allows us then to understand the message of the speech.

Speech recognition can be related to many areas of science: automatic language processing, linguistics, information theory, signal processing, neural networks, artificial intelligence, etc. Among the various fields of research in this area, speech recognition, as well as speech synthesis, speaker identification or speaker verification, are among the most important speech processing techniques.

1.4 Examples

The GAFAM have well understood the potential of the technology, and work on algorithms for years, which are already applied to their products, such as Siri for Apple, Alexa for Amazon, the Google Home, Cortona for Microsoft ... The fields of application of these tools are enormous: help for people, assistance and presence for elderly or disabled people, a source of information, and a way to store information, ... This could have a direct impact on professions such as court clerks or translators. It is therefore a real technological innovation with a significant impact.

1.5 Limits to overcome

However, the current solutions are not yet satisfying, because there are still important errors in real environments. Algorithms still have problems when the speech is in a noisy environment and when several persons speak at the same time. It's a challenge too, to be sure to construct meaningful sentences when creating the text. In addition, researchers such as W. Minker and S. Bennacef, in their publication Speech and Human-Computer Dialogue, have confirmed the complexity of these models by the great difference between formal language understood by machines, and natural language. In formal language, there are strict and unambiguous rules of grammar, spelling and syntax. In natural language, evolutions can occur in the meaning of words, their use, and the message can be very different depending on the context or intonation. It is very difficult for a machine to perceive this.

2 Methodology

2.1 The traditional ASR pipeline

We have began our study by trying to understand the traditionnal ASR pipeline. The steps of implementation are described by Preethi Jyothi for Microsoft in a video in 2017 [1].

Traditional pipeline is composed of several independent components, essentially breaking down the task of speech recognition into building an acoustic model, a pronunciation model and a language model.

Another characteristic is the decomposition of words into phonemes (distinct units of sounds). This allows to construct words based on their pronunciation.

In the first step, the audio signal is converted into a data frame with acoustic features with a Fourier Transform. Different models are then applied successively to transform the data in text:

Acoustic Model: This model is trained to relate audio data (frequencies, intensity with time) to phonemes. This model estimates the probability $P(X|H)$ of the observed sequence of acoustic vectors X given a possible pronunciation H of a given word sequence. [3], [4].

Pronunciation Model: This model is able to create words from an association of phonemes. For example, given the phonemes [g], [uh], and [d], the model is able to understand that it's the word good. This model gives for each sequence of words W, the possible pronunciations H with their probabilities $P(H|W)$ [5].

Language Model: It creates meaningful sentences from an association of words. This model gives

the probability $P(W)$ of each word sequence W in the target language [6], [7], [8], [9].

The problem is to find the sequence of words that has the highest probability. Formally, the problem amounts to finding the sequence of words W that maximises the following function:

$$P(W)\sum_{H} P(H|W)P(X|H) \tag{1}$$

While flexible, this pipeline faces major issues: Since each component is trained independently from the others with different objective functions, error in one block may not behave well with other errors, which makes the task of improving the accuracy particularly difficult. Moreover, building each of these components from scratch requires significant work.

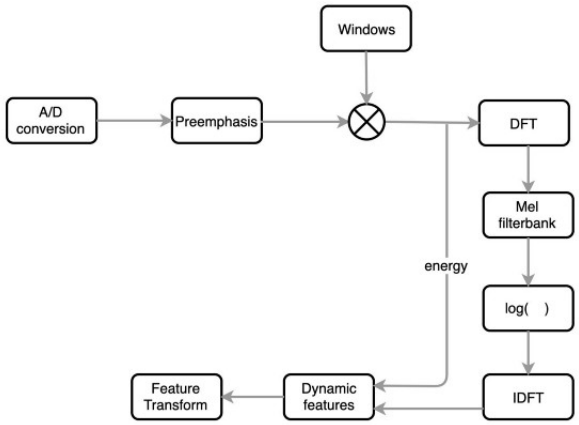
2.2 The first CNN model

2.2.1 The features

The raw audio signal is not suitable to use as input for our model because it contains a lot of unwanted noise. Instead, we can improve the performance of the model by extracting important features from the audio signal and using those as input. One commonly used method for extracting features from audio is called Mel-Frequency Cepstral Coefficients (MFCCs).

MFCC is a feature extraction technique commonly used in speech and audio processing. It converts the audio signal from the time domain to the frequency domain and then applies a non-linear transformation to represent the audio in a way that is more similar to how humans perceive sound. MFCC is a representation of the short-term power spectrum of a sound. The road map of the MFCC computation is the following: MFCC values, which are extracted from audio signals, can be affected by noise. To make them less sensitive to noise, it is common to normalize their values in speech recognition systems. Some researchers have suggested modifications to the standard MFCC algorithm in order to make it more robust to noise.

Figure 1: Road map of the MFCCs computation



The resulting coefficients, which are a set of numbers, can be used as input to a machine learning model for tasks such as speech recognition, speaker identification, and music genre classification.

2.2.2 The model

The model consists of a sequence of layers of neurons, each with a goal in the network:

- convolutional layers: responsible for detecting patterns and features in the input data
- pooling layers: used to reduce the dimension of the data and control overfitting
- flatten layer: used to convert the multi-dimensional array output from the previous layers into a one-dimensional array, so that it can be used as input for the dense layers
- dense layer: responsible for making the final predictions based on the output of the previous layers
- dropout layer: Dropout is a regularization technique used to prevent overfitting in deep learning models. It works by randomly dropping out (setting to zero) a certain percentage of neurons during training, so that the model cannot rely on any one feature too much. This helps to reduce

the chance of the model memorizing the training data, instead of generalizing to new data

Classical CNN models are trained using backpropagation and gradient descent.

2.2.3 The test

We have trained our model on Google Colab, which allows access to Kaggle datasets without downloading them locally. We were able to use their computing power to train our model for 6 hours on 2,000 audios. This represents 5h30 of training audios. We then tested the performance on a few test audios, to see if the predictions were good.

2.3 The CTC model

Connectionist Temporal Classification [2], more commonly referred to as CTC, is an end-to-end system introduced to address the alignment problem, which is one of the main issues in speech recognition. This difficulty arises because audio inputs of variable lengths may need to be mapped to the same transcription depending on the speed at which a person is speaking. CTC is nowadays often used in combination with other techniques in speech recognition tasks.

2.3.1 Key characteristics

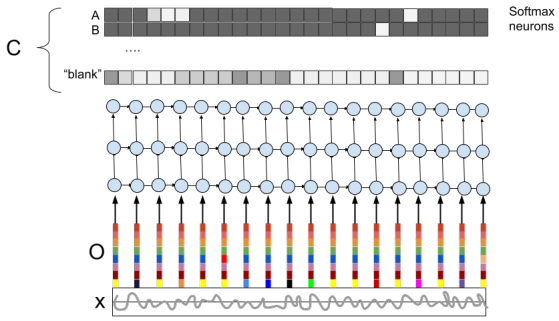
This model is often implemented as a grapheme based model, i.e. outputting characters directly, while traditional pipelines were based on a phoneme decomposition of words. In addition to the letters of the alphabet (including accents depending on the language), the model uses a blank character (denoted by an underscore _ below) and an unknown character.

2.3.2 Architecture

The architecture relies on a Recurrent Neural Network, trained on labelled data (i.e. transcribed audio):

1. Raw audio is pre-processed to obtain a simple spectrogram (which correspond to calculating Fast Fourier Transforms on small 20 ms windows of waveform) to get frequency content.

Figure 2: CTC Architecture



2. Frames go through the RNN. Output neurons of the RNN are softmax neurons encoding a distribution $P(c_i|x)$ over symbols. Under independence assumption,

$$P(c|x) = \prod_{i=1}^{N} P(c_i|x) \quad \text{with } c_i \in \{A, B, ..., \text{blank}\}$$

Note that c is a sequences of symbols that is of the same length as the audio.

3. A mapping β takes as input the encoding c and returns the transcription y. β essentially removes duplicates and blank characters: For example, $\beta(HHH\_EE\_LL\_LO) = HELLO$. Since several c can lead to the same y, the distribution over transcriptions reads

$$P(y|x) = \sum_{c:\beta(c)=y} P(c|x)$$

Network parameters are updated with backpropagation and gradient descent to maximize likelihood of choosing the correct label

$$L(\theta) = P(y^{*(i)}|x^{(i)}) = CTC(c^{(i)}, y^{(i)})$$

2.3.3 CTC in recent works

While CTC as an end-to-end system described above is no longer the state-of-the-art, CTC loss is the ob-

jective function used for fine tuning on labeled data proposed by [3] to perform speech recognition.

2.4 Wav2Vec2.0

Wav2Vec 2.0 is a state-of-the-art end-to-end audio pre-processing technique used for speech and audio recognition tasks. It was published in 2019 and it is an extension of the original Wav2Vec model, which was developed by Meta. The scientists behind the model are Alexei Baevski, Henry Zhou, Abdelrahman Mohamed and Michael Auli.

2.4.1 The advantages

This model has many advantages over older models. First of all, the model is based on a self-supervised algorithm. It therefore does not require supervised data for training. This is extremely beneficial in the case of ASR as it allows the learning of many languages, which was not previously possible. Indeed, in previous models, a lot of audios were needed with their transcription, which is not available in large quantities for all languages except English. For Wav2Veq2.0, the model needed 50,000 hours of audio for training, which would have been difficult with the transcripts. In a second phase, very little labelled data with the transcripts is needed to get greatly improved results: 10 minutes of labelled data was enough.

In addition, there are many packages available today that make using Wav2Vec2.0 quick and easy. We were able to use HuggingSound from HuggingFace, and the implementation was very fast.

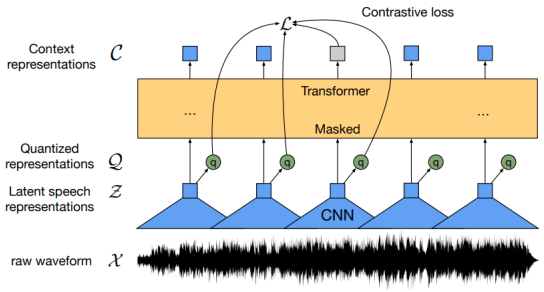
2.4.2 The architecture

Wav2Vec 2.0 is a self-supervised model, meaning it does not require labeled data for training. Instead, it is trained on a large dataset of unlabeled audio files, and learns to extract useful features from the audio by predicting the next audio frame in a sequence. This is done using a Contrastive Predictive Coding (CPC) technique. It is a technique used to learn a compact and semantically meaningful representation of the data. This is done by training a model to predict the next frame of an input sequence. The CPC technique consists of two main components: an encoder and a predictor. The encoder processes the input sequence and generates a fixed-length representation of the sequence, which is called the "context vector". The predictor then tries to predict the next frame in the sequence using the context vector as input.

There are three main steps in the construction of the algorithm:

- CNN with 7 layers to learn the latent speech representations Z
- Quantization: try to match the Z with codebooks, sort of phonemes
- Transformers to learn contextual representation

Figure 3: The Wav2Veq2.0 architecture



The feature encoder is based on a temporal convolution followed by a GELU activation function.

For the training, noise is added to the audios so that the model is able to distinguish the speech and the noise. It really improve the performances of the algorithms in all context. The loss function to minimize is then the composition of 3 functions:

$$Loss = L_{contrastive} + L_{diversity} + L_{L2} \qquad (2)$$

where

- $L_{contrastive}$ measures the similarity between the predicted next frame and the actual next frame in the sequence. It is calculated as the dot product of the predicted

frame and the actual frame: $L_{contrastive} = -\log(\exp s(x, x')/(\sum \exp s(x, x')))$, with s the similarity function, often the dot products

- $L_{diversity}$ is used to encourage the model to generate a wide variety of different outputs, rather than simply memorizing the training data. It measures the dissimilarity or diversity between the generated samples and the real samples
- L_{L2} is used to prevent overfitting in machine learning models by adding a term to the loss function that penalizes certain model parameters. Also known as Ridge regularization, it adds a term to the loss function that is proportional to the square of the model weights. It tends to produce models with small but non-zero weights.

Finally, for the encoder, two language models were trained: a 4-gram model coming from NLP methods and a transformer.

2.4.3 The training

The model was trained on CommonVoice, which is the biggest dataset with audios and transcriptions in French. Then, two pre-trained were created: a BASE (with a dimension of 768) and LARGE (dimension of 1024).

As said in the advantages part, the model was trained on 53,000 hours of unlabelled samples and only 10 minutes of labelled data were needed. For that training, the teams from Meta used the LibriVox and LibriSpeech datasets

3 Evaluation

3.1 The dataset

To test our algorithms and Wav2Vec2.0, we searched for a French dataset with audios and transcriptions. We chose the Kaggle dataset by Kyubyong Park and Tommy Mulc named "French Single Speaker Speech Dataset" [4]. This dataset has 8,600 audios of about ten seconds each, which represents 24 hours of recording. The recordings are of single people reading sentences from the novels Les Misérables by Victor Hugo and Arsène Lupin contre Herlock Sholmes by Maurice Leblanc. We are therefore in the easiest case, where there is no noise or several people speaking at the same time.

3.2 The relevant metric

The most common metric in speech-to-text is the Word Error Rate (WER), which measures the difference between the predicted words and the actual transcription. It takes into account substitutions, insertions and deletions. Thus, we can define the WER by the equation :

$$WER = \frac{S + I + D}{N} \tag{3}$$

where

- S = number of substitutions (word replaced by another in the prediction)
- I = number of insertions (word added in the prediction)
- D = number of deletions (word removed in the prediction)
- N = number of words in the reference

Thus, the WER value varies between 0 and can exceed 100%, because the number of insertions is not limited.

3.3 Traditionnal methodology and results

When testing methods, it is always important for this kind of study to try on a variety of data. Thus, researchers often test on clean data and on noisy data.

Here are some average results for the WER:

- read texts (voice dictation, single-talker system): $WER \approx 5\%$
- radio and TV news: $WER \approx 10\%$
- informal telephone conversations: $WER \approx 40\%$

3.4 Best current models

It exists different models use nowadays, and to compare their performance, it's important to test them on different datasets. Indeed, the environment, the noise, the number of speakers, etc. can largely influence the result. In their analysis published on GitHub on Februaru 2022 [5], Picovoice (one actor of ASR) try to evaluate and compare the performances of the different main algorithms. For that, they average the results on 4 datasets: LibriSpeech clean, LibriSpeech other, TEDLIUM and CommonVoice. Here are the results they obtain for such algorithms:

- Azure: $WER = 7.93\%$
- Amazon: $WER = 8.74\%$
- Google Enhanced: $WER = 11.32\%$
- Google: $WER = 20.46\%$
- IBM: $WER = 22.04\%$
- Mozilla DeepSpeech: $WER = 22.86\%$

3.5 Results

3.5.1 The CNN model

Concerning the CNN model, it was very difficult to train it with our processing capacity. In fact, the model almost always predicts the same words and sentences, and was not too large or not enough trained to have better results. So, the WER was very bad, about 80%. For this type of model with more training, we found in the literature WER above 20% [6].

3.5.2 The CTC model

Starting from [7] we tried to implement our own CTC model but were unable to do so due to insufficient processing power. According to [7], training for more than 50 epochs would be necessary to obtain acceptable results, and each epoch would take 5 to 6 min to train with a GPU (which we don't have). Under these assumptions, they achieved a 16% WER.

3.5.3 Wav2Vec2.0

Concerning Wav2Vec2.0, the last results published were very impressive, and improve a lot the WER comparing to the other models. In the first paper, the authors announced WER of 5.7% on clean audio and 10.1% on noisy audio. In the last paper, the results were even better: they reached 1.8% WER on the test-clean subset and 3.3% WER on the test-other.

We wanted to compare that results and compute our own results. We used Wa2vec2.0 pre-trained models and tried to predict the transcription of 580 audios of our dataset. That represents about 1h30 of transcriptions. With that, we obtain a WER of 21.7%. The results are not as good as in the paper, but in large part because of the punctuation. The Wav2Vec2.0 algorithm predicts sometimes punctuation, such as commas and apostrophes, but not always. We couldn't thus decide whether to take the punctuation into account or not, and it increases the difference between the predictions and real transcriptions.

4 Conclusion

In our study, we were able to familiarize ourselves with traditional ASR methods and seek to understand state-of-the-art algorithms. We were also able to develop a relatively simple CNN algorithm.
We tested this algorithm on the Kaggle dataset. Our results are logically worse, largely because it was difficult for us to train our models for a long time and on a lot of data.
The Wav2Vec2.0 model was very impressive in our tests, as it has very good results and is very easy to implement given . The variety of languages available is also a real advantage. However, the transcriptions are slow (10 seconds for a 10- second audio), which can be a hindrance to obtaining large audio transcriptions.

5 References

[**1**] Microsoft Conference https://www.microsoft.com/en-us/research/video/automatic-speech-recognition-overview/

[**2**] Graves, Alex Fernandez, Santiago Gomez, Faustino Schmidhuber, Jürgen. (2006). Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. ICML 2006

[**3**] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli. wav2vec 2.0: A framework for self- supervised learning of speech representations. In Proc. of NeurIPS, 2020c.

[**4**] Kaggle, Datasets, Bryanpark, French-Single-Speaker-Speech-Dataset https://www.kaggle.com/datasets/bryanpark/french-single-speaker-speech-dataset

[**5**] ASR algorithms benchmark, Github, Picovoice, Speech-to-text-benchmark

[**6**] Ilias Papastratis. 2021-07-14. TheAiSummer, Speech Recognition: a review of the different deep learning approaches WER of CNNs model

[**7**] Mohamed Reda Bouadjenek, Ngoc Dung Huynh, Automatic Speech Recognition using CTC, 2021 https://keras.io/examples/audio/ctc_asr/