Automatic Speech Recognition Using Deep Neural Networks Trained with Self-Supervised Approaches

Abstract

Research on automatic speech recognition has seen great advances and changes in recent years. Building on a long history of research both in signal processing and in statistical learning, deep neural-network based systems now achieve performances on par with, or nearing that of human transcriptors fluent in the language at stake. Most recently, multiple approaches using self-supervised learning have come forward, with results defining the current state-of-the-art in speech recognition systems. One crucial component in these systems is their ability to leverage large quantities of unlabelled data to build meaningful representations of speech signals.

This internship aims at building a deep understanding of these models, and exploring to what extent these allow for improvements in speech recognition across languages with few resources. Self-supervised ASR models usually require two phases of training, which are both computationally expensive and highly dependent on the language at stake. This report highlights the specificity of each of these phases and their contribution to the performance of the model, and compares mono-lingual approaches (using data from the target language only) to multi-lingual ones for the case of ASR in French. Additionally, a thorough analysis of the hyperparameters of the models is provided, along with intuitions on the dependence of the performances on these parameters. In short, the conclusions of the report are that multi-lingual models significantly outperform mono-lingual approaches in our research, that language models significantly increase the quality of the transcriptions both in French and English, and that the phase of pre-training plays a role in the correctness of the transcriptions that is all the more important that the fine-tuning phase uses little labelled data.

Acknowledgments

I would first like to express my utmost gratitude to my internship supervisors for their kindness, patience and immense knowledge, and for their continuous support in helping me better understand and solve issues that I encountered in my research. Besides them, I would like to thank the team in which I am grateful to have been involved during this internship, both for their guidance on scientific endeavours as well as for their stimulating discussions and sense of humour. Last, but not least, I would also like to thank the teachers of the MVA master degree courses that I attended, for their dedication in making their lectures interesting and open to all, and for allowing me to pursue an internship on these topics.

Table of contents

1 Introduction . . . 1

2 Related work . . . 2
2.1 Definition of the task at hand . . . 2
2.2 A brief historical overview of ASR systems . . . 3
2.3 End-to-end models and recent advances . . . 7
2.4 Low-resource languages and multi-lingual models . . . 16
2.5 Domain adaptation for ASR models . . . 18
2.6 Other unresolved tasks and open questions in ASR . . . 21

3 Resources and experimental setup . . . 22
3.1 Data and languages . . . 22
3.2 Training procedure and modules . . . 23
3.3 Technical challenges . . . 38

4 Using end-to-end backbone ASR models in practice . . . 39
4.1 Pre-training mono-lingual ASR models . . . 39
4.2 Fine-tuning mono-lingual ASR models . . . 43
4.3 Language modelling and decoding . . . 45

5 Adapting ASR systems across languages . . . 50
5.1 Universal, cross-lingual or specific modelling . . . 50
5.2 Mono-lingual ASR models . . . 51
5.3 Fine-tuning multi-lingual ASR models . . . 58
5.4 Pre-training multi-lingual ASR models . . . 61

6 Discussion and limitations . . . 62
6.1 Results and limitations . . . 62
6.2 Decoding challenges . . . 63
6.3 Further work . . . 63

7 Conclusion . . . 64

A Appendix . . . 65

Figures . . . 76

Tables . . . 78

B Bibliography . . . 79

1 Introduction

Interdisciplinary research on automatic speech recognition (ASR) systems has recently benefited broadly from advances in the field of deep learning and the development of neural networks for sequence-to-sequence modelling, as well as from the increased availability of large datasets of transcribed, and non-transcribed speech audio recordings. In short, the task of ASR, which consists in enabling the recognition and transcription of spoken language into text by a computer-based automatic algorithm (with the main benefit of searchability through transcribed data) now is on par, or outperforms transcriptions made by human non-expert transcriptors on clean data in English. Although vastly contingent on the recent use of sequential neural networks, these advances in ASR are also highly dependent on a long history of discoveries and improvements made in speech audio analysis, and, to a larger extent, in signal processing.

As its first ambition, this internship report covers in depth some of these recent advances, with a particular focus on the operating principles of state-of-the-art the wav2vec2 and XLS-R models introduced by Baevski et al. (2020) and Babu et al. (2021) respectively, and tries to situate these systems in the long-term history of automatic speech recognition. Then, this report's second aim is to experiment with these models, and explore their limitations with regard to their development for real use-cases, with a substantial analysis of their use on data in other languages and in lower quality. In particular, it experiments with the impact of modifications during the training phases of these models (pre-training and fine-tuning) when learning on French audio data and when using different amounts of labelled data for the fine-tuning phase, both in English and in French.

The contribution of this report is thus threefold. First, it presents a comprehensive overview of the history of ASR, from the first edeavours of the 1950s to the most approaches, shedding light in particular on a number of recent advances in end-to-end systems in speech processing. Then, it aims at producing a detailed account and analysis of the working-horse models that are the wav2vec2 and XLS-R models, with a thorough review of the internal architecture of these models, the functioning of self-supervised approaches in ASR, and the dependence of the performances on the data and choice of hyperparameters. Finally, it explores these models in-depth experimentally, by providing results regarding all training phases (pre-training, fine-tuning and decoding), with the underlying question of the need to train these models from the start if switching languages and data domains.

The first conclusion of this report is that the training of these models is an arduous task both on the scientific and technical levels. As a matter of fact, the training procedures require expensive computational resources as well as a thorough analysis of the influence of hyperparameters on the convergence properties of self-supervised sequence-to-sequence deep learning approaches. Then, although research on this question is still necessary, this report aims at emphasising that mono-lingual approaches in end-to-end self-supervised systems do not necessarily outperform multi-lingual approaches, and that the differences in performance between the two does not justify a specific pre-training procedure for each language. On the other hand, this report showcases that the amount of labelled data plays a significant role in the performances of the model, particularly with multi-lingual models, and that, as a rule-of-thumb, a corpus of about 10 hours is necessary for high-quality transcriptions.

2 Related work

ASR systems have seen great improvements in the last few decades. Like in many other fields, the first efficient use cases of deep feed-forward neural networks for ASR were introduced in the early 1990s, confirming a long-standing belief that neural networks would fit well the requirements of acoustic modelling tasks (Hinton et al. 2012). Building on these advances, fully end-to-end (E2E) architectures have been shown to achieve human accuracy over a variety of downstream tasks in English (Baevski et al. 2020; Hsu, Bolte, et al. 2021; S. Chen et al. 2021). Yet, current ASR systems still fall short of many challenges, and speech recognition across languages or domains are still very much open research questions.

In the following section, we propose a brief overview of ASR and the research developments that led to the current state-of-the-art speech recognition systems, along with an outline of unresolved issues in ASR and the recent literature that tries to address them.

2.1 Definition of the task at hand

Automatic speech recognition (ASR) consists in the task of transcribing spoken language into text with the help of an automated, computer-based system, with the main aim of rendering easier the search of content within audio data. Usually, ASR is performed on digitally recorded audio data containing speech, and the desired output takes the form of sequences of characters.

In non-rigorous terms, one can say that the main desirable property in the task of ASR for a specific input and desired output is that the text transcribed from the audio data matches sufficiently well the transcription that would have been made by a person fluent in the language at hand. As such, ambiguities can exist, such as when an isolated word recorded in a noisy environment cannot clearly be transcribed, even by humans. In the following, we ignore these ambiguities in the labelled data and consider the ground-truth transcriptions to be faithful. More importantly, defining metrics that measure the quality of the transcription is an arduous task, which is itself an active research question (Papineni et al. 2002; C.-Y. Lin 2004). In the following, we rely mostly on the most common evaluation metrics, such as the word error-rate (WER), character error-rate (CER) or phoneme error-rate (PER) that are defined in Section 3.2.

In a more abstract and mathematical formulation, the problem of ASR can, like many tasks, be described as an optimisation problem. To do this, let $T \in \mathbb{N}$, and $X = (x_1, ..., x_T)$ be a digitally stored audio recording of length T with $\forall i \in \{1, ..., T\} : x_i \in \mathbb{R}$. Let also $N \in \mathbb{N}$ and $Y = (y_1, ..., y_N)$ be the sequence of written language units (*i.e.* characters or graphemes) corresponding to the transcription of the audio recording. Denote also by $\mathcal{Y}$ the set of all possible sequences of written language units for a given dictionary. Then, the problem of ASR can be written as the following optimisation problem:

$$\hat{Y} \in \underset{Y \in \mathcal{Y}}{\operatorname{argmax}} \ \mathbb{P}(Y \mid X)$$

In words, the problem of ASR can be written as finding the sequence of written language units with maximum posterior probability given an input sequence of audio recordings of given length, among all possible sequences of written language units.

2.2 A brief historical overview of ASR systems

Attemps to turn speech data into text using automated computer-assisted tools have been carried out since the 1950s. Like in computer vision, the first such tentative works were aimed at recognising digits. Davis et al. (1952) were the first to propose an automated system, which was able to distinguish between digits spoken at a normal speech rate by a speaker once the system had been adapted to that speaker's voice. In essence, the system worked by searching for the closest pattern among manually coded references consisting of multiple discretised frequency bands of the recording's spectrum. Using a comparable system, Olson et al. (1956) built a system to identify which of 10 syllables were pronounced by a single speaker. In a similar fashion, but already with the far-sighted idea of generalised speech recognition, Forgie et al. (1959) developed the first vowel recognition computer program using live spectral data from speakers and similarity scores between the two first formants of the recorded data, as well as pre-analysed speech spectral data from the same speakers. Building on these seminal developments, several laboratories in Japan and in the U.S. went on proposing highly specific hardware related to speech recognition in the 1960s: for Japanese speech recordings, Suzuki (1961) were the first to introduce a vowel recogniser, Sakai et al. (1961) a phoneme recogniser, Nagata (1963) a digit recogniser; and for speech recordings in English, Denes (1959) presented the first phoneme recogniser with a statistics-driven pattern recognition mechanism.

Although pioneering, these approaches were short of two desirable properties of ASR systems: maintained efficiency across multiple speakers, and non-specific transcription of speech content. The 1970s saw multiple major breakthroughs to address these shortcomings. Importantly, one crucial obstacle to the aforementioned works was the non-uniform time scale of speech. Across speakers, but also for a single speaker, speech varies greatly in length. Some phonemes take longer to be pronounced, and even the same phoneme can be vocalised differently. Martin et al. (1964) was the first to recognise the need to address the temporal non-uniformity in speech utterances by adding an utterance endpoint detection system on top of a speech recogniser. Then, building on the work of Bellman (1954), multiple research groups started applying dynamic programming with pattern recognition methods to automatic part-of-speech recognition tasks. In short, dynamic programming allows one to compare efficiently two time series of different lengths, and to give a measure of the distance between them. Hence, when applied to speech recognition, and coupled with a pattern recognition algorithm, dynamic programming allows one to efficiently match recorded speech utterances together, despite them varying in lengths. Vintsyuk (1968) was the first to propose using dynamic programming for speech alignment, followed quickly by Velichko et al. (1970), Sakoe (1971) and later on Sakoe and Chiba (1978) who formalised the idea into the concept of dynamic time-warping for speech pattern matching. Since then, numerous variants have emerged, some of which are still being used today in speech recognition systems' pipelines, including, notably, adaptations of the Viterbi algorithm (Viterbi 1967) for decoding in speech-to-text tasks.

Systems built in the 1970s and early 1980s also benefited largely from more general advances in speech and language analysis and processing. Following the formulation of Linear Predictive Coding (LPC) by Atal et al. (1971) and Ikatura (1970) for the estimation of formant frequencies, pattern recognition systems started relying on LPC methods for matching speech utterances (Ikatura 1975; Rabiner et al. 1979). Notably, more discriminative and compact speech processing

features were introduced, in particular mel-frequency cepstral coefficients (Mermelstein 1976). Likewise, the early development of language models, with syntactic and word boundary rules, and of graph-search algorithms allowed to integrate language knowledge in the recogniser systems. Although the term was coined later on by Reddy et al. (1977), the beam-search algorithm was introduced by Lowerre et al. (1976), with the goal of searching for the highest matching score within predicted sequences that match some language knowledge constraints. Its development took place within the Harpy speech recognition system for the Advanced Research Projects Agency of the U.S. Department of Defense, which itself was the first word-recognition model to use a language knowledge in decoding, and which used a vocabulary of about 1000 words in English. In line with this, Jelinek et al. (1975) were the first to integrate an n-gram model in the decoding part of the system. In short, as detailed in section 3.2, n-gram models, which are still widely used today, are simple language models which model grammar statistically by characterising the likelihood of occurences of spans of n words. As such, it allows to improve the efficiency of the ASR system by correcting for errors using implicit knowledge about language grammar.

Another milestone in the development of ASR systems came from the introduction of hidden Markov models (HMM) by Baum and Petrie (1966), among other research groups. Thus far, speech recognition systems relied heavily on template-based approaches, where utterances (words, digits, etc.) were implemented one by one to be used later in pattern recognition algorithms. Although acknowledged early on by laboratories working on automatic speech recognition (*e.g.* Baker (1975)), HMMs only started to be used substantially in the 1980s following the influential works of Ferguson (1980) and Levinson et al. (1983). In broad terms, hidden Markov models allow to take into account the intrinsic variability of speech signals as well as the structural properties of language in a single statistical framework by estimating the parameters of a doubly stochastic process. More specifically, a standard HMM consists of a Markov chain, which accounts for an unknown phonotactic structure of the transcription, and a set of probability distributions, each associated to a state in the Markov chain, that model the variability in the speech signal of the spoken utterances. As such, it allows to model in a probabilistic way sequences of utterances that are not fixed but rather change according to the transition probabilities of a Markov chain. One major step forward with using HMMs is also found in the automation of parameter estimation for pattern recognition. Where, in most past systems, matching between utterances was performed on carefully designed sets of speech features, HMMs can be trained using the Baum-Welch algorithm (Baum et al. 1972), which allows for estimation of the probability distributions and Markov chain parameters using a training set of utterances of interest and corresponding transcriptions.

The 1980s also saw the first practical uses of artificial neural networks (ANN) for automatic speech recognition. ANNs were introduced theoretically for the first time in the 1940s with the work of McCulloch et al. (1943), and in the form of the perceptron in the 1960s by Rosenblatt (1958), but failed to be employed efficiently in applications at the time. As for many other tasks, the earliest attemps to use ANNs for automatic speech recognition were carried out in the late 1980s and early 1990s. In 1988, Elman et al. (1988) were among the first to use a neural network trained by backpropagation for speech recognition, and showed the potential of neural architectures by achieving state-of-the-art classification results on pre-segmented speech utterances. In a another attempt, Waibel et al. (1989) demonstrated the strenghts of neural

networks based systems for speaker-dependent recognition by improving over the state-of-the-art of the time by a considerable margin (from 93.7% with a HMM to 98.5% accuracy with their method). Notably, their approach was one of the first to try to address the issue of time variability with neural networks, by using a time-delay neural network (TDNN) consisting of interconnected linear layers accounting for shifts in the endpoints of the speech utterances. In regard to this, Lippmann (1989) gives a detailed account of the early research on the use of ANNs for automatic speech recognition. Yet, ASR in more complicated use cases inevitably requires some management of the temporal variations in speech, which neural networks in their original form or feed-forward neural networks failed to address. In fact, until around 2010, state-of-the-art models for ASR had almost all been using mixtures of HMM-based Gaussian densities with mel-frequency cepstral coefficients as input features, and it is only with the presentation of novel architectures that neural networks started to outperform by a large extent HMM-based systems. Yet, as of today, many systems still rely on HMMs to perform speech processing tasks, including systems that interchange GMMs with ANNs such as the one proposed by Bourlard et al. (2012).

Pioneering research on the use of neural networks for ASR stem directly from the introduction of recurrent neural networks architectures by Rumelhart et al. (1986) and the vast research that followed it. In short, recurrent neural networks are adaptations of feed-forward neural networks where connections between nodes can be made along a temporal sequence, hence allowing to model temporal dynamic behaviours in the input features in a way that traditional ANNs can not do. As such, their properties are well-adapted to the task of speech recognition, where input features are sequences where contextual information is crucial in prediction tasks. However, applicability of RNNs have long been limited by the fact that traditional use of RNNs for speech data requires pre-segmented speech data, which remains an ubiquitous problem for real-world data. In fact, the standard objective function in traditional RNNs require separate predictions for each point of the training sequence, implying that the output can only consist in independent label classifications, thus requiring some considerable pre-processing. Hence, for a long time, the most effective use of RNNs for ASR has been that of combining a RNN with a HMM in a way that lets the HMM model the long-range dependencies.

Proceeding from this, one of the most influential works in the building of current state-of-the-art ASR systems has been the development of Connectionnist Temporal Classification (CTC) by Graves, Fernàndez, et al. (2006), which addresses the temporal dependence issue in using RNNs. In short, as described in section 3.2, CTC is a differentiable objective function that relies on interpreting a RNN output as a probability distribution over all possible sequences of labels conditioned on the sequential input features, which removes the need for pre-segmenting data.

The use of recurrent neural networks, and later on long short-term memory neural networks (Hochreiter et al. 1997) along with connectionnist temporal classification have fundamentally changed the approach taken by research on automatic speech recognition systems. Already in their presentation of the CTC objective function, Graves, Fernàndez, et al. (2006) demonstrate a significant improvement of their RNN system over systems using context-independent and context-dependent HMMs, with a label error rate on the TIMIT dataset lowered from 35.2% to 30.5%. Not much later, Eyben et al. (2009) demonstrated the use of the CTC loss function alongside an LSTM neural network by building one of the first automatic speech recognition systems to output a distribution probability over graphemes (*e.g.* letters) rather than acoustic sub-word units like phonemes. These advances have greatly improved the performances of general

automatic speech recognition systems. In their influential comparison study of traditional HMM models with RNNs, Graves and Jaitly (2014) document a 27.2% absolute difference in terms of WER between a baseline HMM model and a RNN transcription model with CTC objective function when using only dictionary matching as language model.

The models presented above are different in roughly three ways from current state-of-the-art ASR systems. First, as noted by Graves and Jaitly (2014), up until roughly 2016, most approaches taken with RNN models were not entirely end-to-end in the sense that the features taken as input consisted in already pre-processed speech features such as mel-frequency cepstral coefficients or filterbanks. Additionally, due in part to computation resources limitations, the first RNN-based models did not leverage the large quantities of existing labelled data (for instance, from audiobook corpora). In that regard, the work of A. Hannun et al. (2014) on DeepSpeech, and later of Amodei et al. (2016) on DeepSpeech 2 were breakthroughs in exploiting large amounts of data from varied sources in training an end-to-end model (about 5000 hours of labelled read speech from about 9600 speakers). DeepSpeech was also one of the first approaches to use spectrograms as input features rather than MFCCs or other preprocessed acoustic features, paving the way for fully end-to-end systems. Moreover, leveraging High-Performance Computing (HPC) for training, DeepSpeech 2 achieved state-of-the-art results in terms of ASR on English and Mandarin (with respect to UERs on Librispeech and Baidu corpora), beating for the first time human annotations. Up until about 2019, DeepSpeech 2 remained the best performing model on common evaluation datasets in English.

Then, perhaps most importantly, until 2019, all ANN-based ASR systems relied on using manually labelled data, but did not leverage in any way unlabelled speech data, which naturally is available in much greater quantities. Inspired by the BeRT achitecture of Devlin et al. (2018) in the field of Natural Language Processing (NLP), multiple teams simultaneously proposed to use self-supervised learning methods for training ANN-based ASR models. In short, self-supervised learning is a two-step learning method in which in the first step, weights are updated using pseudo-labels, for instance by leveraging a masking scheme to hide some information and train the model to retrieve it, and in the second step, supervised learning is used to fine-tune the weights of the model to a task. This two step process allows using unlabelled data during the first part of the training scheme, which is particularly useful in contexts where a lot of unlabelled data is available like in text and speech. In broader terms, self-supervised learning helps in obtaining meaningful latent representations of the data that can later be used as an efficient anchor point when solving a supervised downstream task. The next section presents in greater detail these models, which attain state-of-the-art performances as of today.

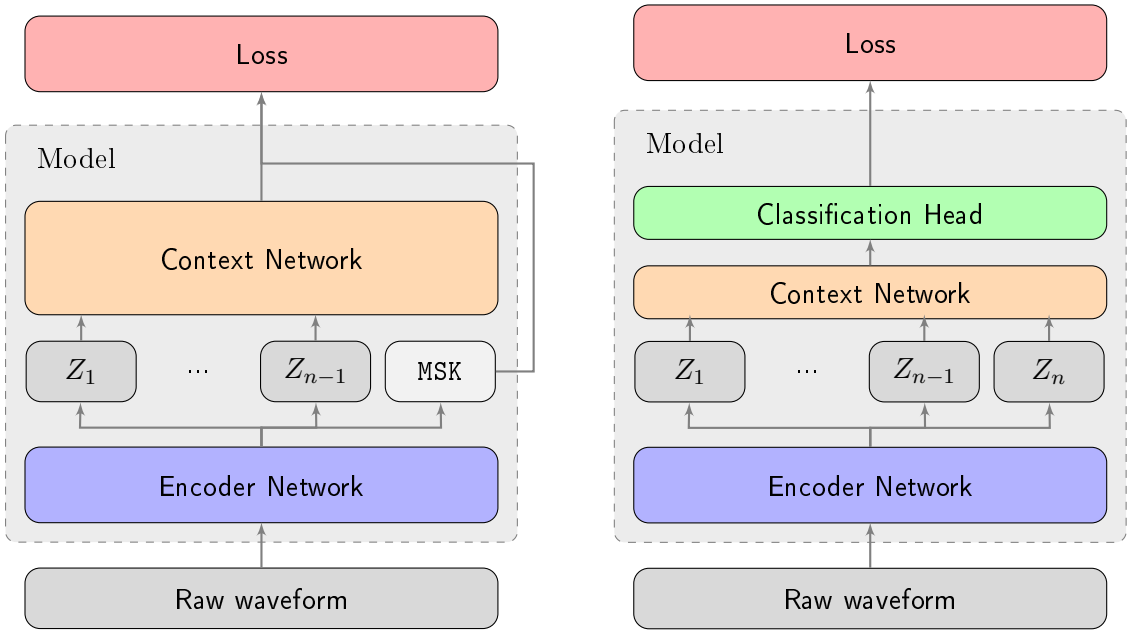
In short, the last few decades have seen both the birth of the first ASR systems and of end-to-end self-supervised models outperforming human speech recognition in English. Like many tasks in computer vision or natural language processing, improvements in speech recognition technologies have gone from manually designed features to recognise simple elements, like digits, to large pipelines combining learning-based modules and hand-crafted features, and, more recently, to fully end-to-end systems taking raw speech signals data as input with little to no pre-processing. Today, ASR systems are used in many settings, including individually by users for dictation or for smartphone assistants, and in public utility settings like automated transcription of speech in movies. While many challenges remain, the field is moving fast and crucial improvements are being made, not only in terms of performance but also in terms of deployment.

2.3 End-to-end models and recent advances

As highlighted in the previous historical section, up until roughly 2019, end-to-end ASR models did not leverage unlabelled data for automatic speech recognition tasks. Some early works on unsupervised representation learning methods for speech were presented as early as in 2016 (Synnaeve and Dupoux 2016; Chung, W.-H. Weng, et al. 2018), but were not exploited for ASR until the breakthrough work of Schneider et al. (2019) and the wav2vec model of contextual speech representations, which attained state-of-the-art performances at the time.

Since then, substantial improvements have been made on ASR systems building on the original idea of Schneider et al. (2019) and leveraging unlabelled data, including, notably, the follow-up work of Baevski et al. (2020) with wav2vec2, along with the Conformer architecture of Gulati et al. (2020) and the HuBeRT model of Hsu, Bolte, et al. (2021). More generally, the advances in terms of representation learning allowed the introduction of models aimed at solving multiple speech processing downstream tasks, like the recent WavLM model of S. Chen et al. (2021). This section covers these models architectures and performances in-depth, starting by an overview of the original wav2vec model and building on model complexity afterwards.

Broadly speaking, end-to-end models using self-supervised learning approaches are based on two phases corresponding to two model architectures, as displayed in Figure 1.



(a) Pre-training phase (b) Fine-tuning phase

Figure 1: End-to-end two-phases self-supervised model dual architecture for ASR

In summary, wav2vec introduced unsupervised representation learning for speech data but still relied heavily on acoustic models, while wav2vec2 and HuBeRT leverage unsupervised learning of contextual representations as part of a fully end-to-end speech processing pipeline. More recently, Conformers and the WavLM architecture have shown improvements by altering parts of the model, but still building on the same E2E pipeline introduced by wav2vec2.

wav2vec The wav2vec model introduced by Schneider et al. (2019) is the first model to have applied unsupervised training to ASR. Although it was pioneering in that sense, the approach builds on a long history of advances in unsupervised representation learning. To give some perspective, unsupervised pre-training had already been shown to improve the accuracy of classification models in computer vision as early as in 2015. In a seminal work in that regard, Doersch et al. (2015) showed that meaningful visual features could be obtained via unsupervised training on patches, allowing to perform unsupervised visual discovery of objects in images. Perhaps more importantly, wav2vec inspires itself from advances made in unsupervised representation learning for natural language processing (NLP), such as the BeRT model of Devlin et al. (2018). In short, BeRT is a pre-training method and model designed to obtain bi-directional semantic representations of text using Transformers (Vaswani et al. 2017) that can be used as features for fine-tuning natural language processing models over a wide variety of downstream tasks, including text generation, question-answering, classification, named entity recognition, etc.

Although continuous speech data is, in many regards, different from text data, the idea of using semantic bi-directional representations for speech processing and modelling was key in the recent advances in ASR. However, in contrast with BeRT, representation learning in ASR pipelines was first introduced using convolutional layers rather than Transformers, which were originally created specifically for text. As a matter of fact, Transformers require discrete data points, which is the case of text represented by a finite dictionary of text units, and not of speech unless a discretisation scheme is operated on the data or its embeddings.

As depicted in Figure 2 below, the wav2vec model consists in two stacked convolutional neural networks. Taking the raw audio waveform as input, the first of these two components is a CNN encoder network which embeds the audio signal in a lower-dimensional latent space. Then, taking as input these embeddings, the second component is a CNN context network, whose goal is to obtain contextualised representations of the embeddings which can discriminate among multiple examples to predict the embedding of a future time frame. To do this, the model is trained using a contrastive loss. In simplified terms, the CNN context network takes n embedded frames as input, and the loss is computed by comparing the similarity of the embedding predicted by the CNN context network at k time steps in the future with the embeddings of the true future time frame. Here, the similarity is measured by comparing the dot product between the predicted embedding, the ground-truth embedding, and a set of distractor embeddings drawn from a carefully set probability distribution characterising speech data in existing time frames.

In more detail and with some mathematical abstraction, let $\mathcal{Z}$ represent the space of embedded features, and $\mathcal{C}$ the space of contextualised features. For $i \in \{1, ..., N\}$, denote by $z_i \in \mathcal{Z}$ the i-th embedded feature frame, and by $c_i \in \mathcal{C}$ the contextual representation produced by the CNN context network taking the n embedded feature frames surrounding z_i as input. Denote additionally by $h_k : c_i \mapsto c_i^\top W_k + b_k$ a step-specific affine transformation, and by σ the sigmoid function. Then, for T time steps, the contrastive loss can be computed as a the sum of a series of losses at each time step $k = 1, ..., T-1$, which can be expressed as:

$$\mathcal{L} = -\sum_{k=1}^{T-1}\sum_{i=1}^{T-k}\Big(\log\sigma(z_{i+k}^\top h_k(c_i)) + \lambda\mathbb{E}_{\tilde{z}\sim p}[\log\sigma(\tilde{z}^\top h_k(c_i))]\Big)$$

In short, one can see from the expression that the loss decreases when the similarity (expressed as a dot product) between the embedded time frame z_{i+k} and the affine transform of the contextual representation $h_k(c_i)$ increases, and increases when the similarity between the embedded time frame z_{i+k} and affine transforms of the distractor embeddings increase. Hence the model helps build context-aware representations that are discriminative for the data at hand. notably, in practice, the distractors are not sampled from a fixed probability distribution (which would be too hard to characterise) but are instead sampled uniformly from audio sequences.

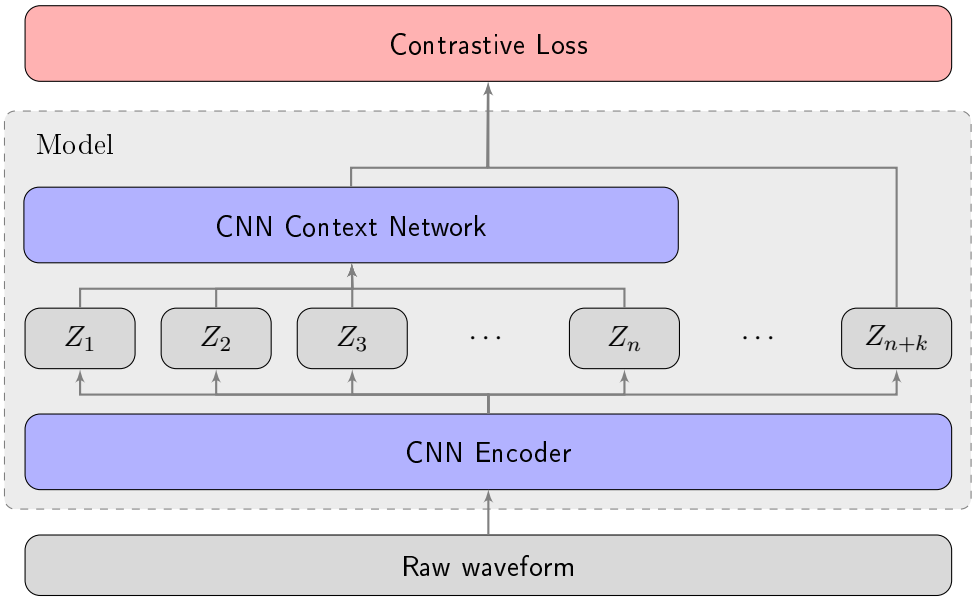


Figure 2: wav2ec pre-training model architecture

Using this architecure, Schneider et al. (2019) were the first to offer contextual representations of speech sequences with the end aim of ASR. Taking these contextual features as input to a variety of acoustic models, wav2vec was shown to achieve a WER of 2.43 on the nov92 test set of the Wall Street Journal (WSJ) dataset, improving by a fair margin on the state of the art of the time of 3.10 achieved by Amodei et al. (2016) with the Deep Speech 2 model. More importantly, Schneider et al. (2019) show that using contextual features from the wav2vec model systematically improves the performance of acoustic models over using log-mel filterbanks.

wav2vec2 The wav2vec2 model introduced by Baevski et al. (2020) directly builds on the wav2vec model but improves on it in two important fashions. First, it simplifies conceptually the speech processing pipeline by integrating unsupervised representation learning and fine-tuning supervised classification of speech time frames in a single deep-learning architecture, like in that of Figure 1. Then, it leverages the attention mechanism (Vaswani et al. 2017) by a discretisation scheme, in order to better integrate contextual information in the representations.

In short, wav2vec2 maps speech time frames to text units (*e.g.* letters or transcribed phonemes) using a model with two main neural networks components, as displayed in Figure 3. First the model encodes raw speech audio data using a multi-layer, fully-convolutional neural network, in the manner of the first encoding CNN architecture in the original wav2vec model.

Then, inspired by the BeRT architecture (Devlin et al. 2018; Jiang et al. 2019) and other masked language modelling techniques, it quantizes the output of the CNN Encoder and masks spans of the resulting quantized speech representations. Then, these latent representations are fed to a multi-layer Transformer neural network to build contextualised representations.

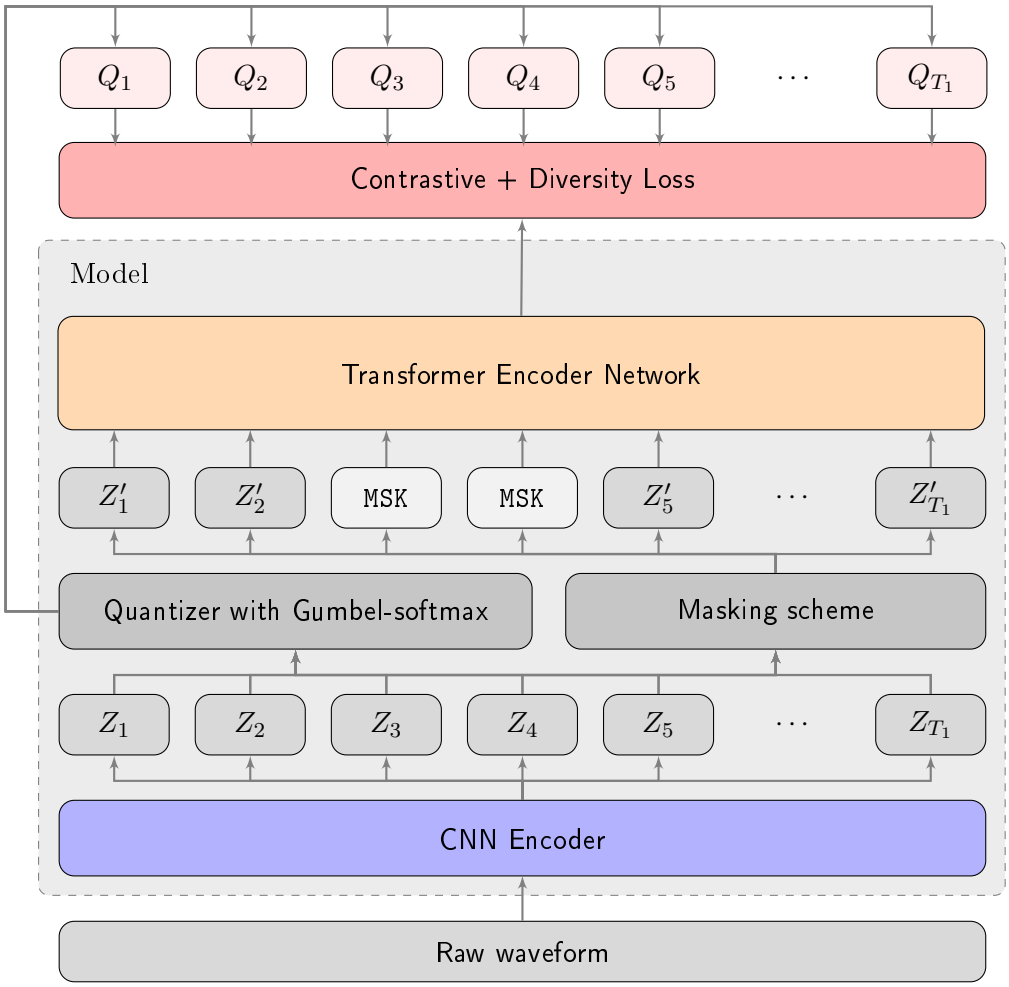


Figure 3: wav2ec2 pre-training model architecture

The end goal of the wav2vec2 model is to work as a fully end-to-end model, mapping directly raw speech data at each time frame to units in a dictionary of text units. To do this, training the model is done in two main phases. First, a pre-training phase, in which the weights of the whole model are updated using a contrastive loss function in the same manner as in the original wav2vec model. This phase allows to build meaningful, contextualised quantized representations of the speech data. Although it now uses a Transformer architecture, similarly to wav2vec, the underlying idea is to build representations via the CNN Encoder and to make them contextualised by a masking scheme during which the aim of the model is to identify the the right quantised representations using the output of the Transformer model. The loss used for pre-training is a weighted combination of a contrastive loss, which measures the model's capacity to identify the right latent quantized representations of masked speech time frames among distractors, and a

diversity loss, which is designed to increase entropy in the use of the quantized representations. Then, once the model has been pre-trained (using unlabelled data), labelled data is used to fine-tune the model during a fine-tuning phase, in which masking and quantizing operations are not carried anymore, a multi-layer perceptron (MLP) is added as classification head and only the Transformer and MLP layers are updated. During this phase, a dictionary of possible text units is defined, and the connectionnist temporal classification (CTC) loss function (Graves, Fernàndez, et al. 2006) is used for the weights updates. More details on the pre-training and fine-tuning phases of the wav2vec2 model are available in Section 3.2.

The wav2vec2 model played a significant role in obtaining state-of-the-art, human-level accurate results in ASR on known corpora. Baevski et al. (2020) test the model capabilities on the LibriSpeech and Libri-light datasets, and on the TIMIT dataset. Using LibriSpeech and Libri-light as training data, wav2vec2 improves over the state-of-the-art of the time by a fair margin. Using the 60,000 hours of unlabelled data from Libri-light for pre-training, and the 960 hours of labelled data from LibriSpeech for fine-tuning, wav2vec2 achieves a WER of 2.2 on the clean test set, and of 4.5 on the other test set with no language model. Using a language model and the same configuration, the WER drops to 1.8 and 3.3 respectively. On the TIMIT dataset, using the same data for pre-training and a 10 hour subset of the TIMIT corpus for fine-tuning, wav2vec achieves a PER of 8.3 on the test set, improving over the state-of-the-art of the time by roughly 29% in relative terms.

Conformer The Conformer architecture presented by Gulati et al. (2020) is another fully end-to-end model for ASR leveraging attention-based modeling architectures. The idea of the Conformer builds directly on the impressive results obtained for NLP and ASR by Transformers, and on their usual combinations with convolutional neural networks (CNN). In essence, a wide variety of research using these models pinpoints the ability of Transformers to capture contextual and global interactions in content, hence, in the field of ASR, at using contextual information to improve the discriminative properties of latent representations and in the end increasing the prediction accuracy of utterances. Conversely, CNNs have been repeatedly identified as efficient in exploiting local dependencies in the data. The Conformer model aims at combining these desirable properties in a single architecture which couples the Transformer and CNN architectures, in order to model both the local and global dependencies in the data at the same time.

Multiple works have identified the importance of combining the information from local and global features early-on in training by combining CNNs and attention-based neural networks: Bello et al. (2019) introduced the attention-augmented convolutional neural networks for computer vision classification tasks, and Z. Wu et al. (2020) combine local context modeling via convolutions and long-distance relationship modeling via attention for NLP tasks. However, Gulati et al. (2020) are the first to introduce a novel way to combine convolutional and Transformer-based modules for the task of ASR.

In short, the Conformer architecture combines self-attention and convolutional modules in a way such that the self-attention learns the global interactions in the data whilst the convolutions capture the relative-offset-based local correlations. To do this, inspired by the Macaron-style network architecture of Y. Lu et al. (2019), Bello et al. (2019) define the Conformer blocks in the manner of Transformer blocks, with two feed-forward modules surrounding a multi-headed self-attention module with relative positional embedding, followed by a convolution module.

Gulati et al. (2020) evaluate the Conformer architecture using the Librispeech corpus. Unlike wav2vec, the original model presented is supervised and does not leverage any unlabelled data in an unsupervised manner, although many adaptations to include a pre-training phase have been carried, including ones that combine the wav2vec architecture with Conformer blocks. When training on the whole LibriSpeech training data, the Conformer architecture achieves a WER of 2.3 on the clean test subset and of 5.0 on the other test subset, results which drops to 1.9 and 3.9 respectively when using a language model, hence coming close to the performances of wav2vec2 with no unsupervised pre-training, hence no contrastive loss.

HuBeRT The HuBeRT model introduced by Hsu, Bolte, et al. (2021) is another fully end-to-end self-supervised model for ASR which takes inspiration from both the BeRT model (Devlin et al. 2018) training process and the original architecture of the wav2vec2 model (Baevski et al. 2020). In short, Hsu, Bolte, et al. (2021) base their approach on the observations that the wav2vec2 model, along with some derivative works, fall short of taking into account three issues: first, each utterance contains multiple sound units; then, although quantization is used, there is no lexicon of input sound units during pre-training; and finally: sound units have variable length with typically no known segmentations. These issues can typically hinder the model's ability to learn meaningful representations during the self-supervised pre-training phase.

As a matter of fact, as highlighted by Hsu, Bolte, et al. (2021), speech data differs crucially from text or images in that speech signals are continuous-valued sequences. Firstly, self-supervised learning for speech cannot take the instance classification assumption typically taken in self-supervised methods for computer vision, because each utterance can potentially contain multiple sounds (where, in computer vision, the label *dog*, for instance, refers to images of dogs which cannot be both a dog and a cat at the same time). Secondly, unlike in NLP applications with word-pieces, self-supervised pre-training for speech cannot use a pre-established finite lexicon of sound units. Lastly, masking schemes are inherently less efficient since the boundaries between sound units are not known a priori in the pre-training phase.

Building on these limitations, the HuBeRT model adds a step to the training process compared to the wav2vec2 model, which consists in discovering hidden units targets through clustering, hence leading to changes in the loss function and rendering the quantization module of the wav2vec2 model not needed anymore. In more detail, as detailed in Figure 4, this first step consists extracting hidden-units (*i.e.* pseudo-targets) from the raw waveform or MFCCs of the waveform, using a clustering algorithm such as the K-means algorithm. Then, each resulting cluster becomes a hidden unit, and all audio frames are assigned to the hidden unit linked with to them. Finally, each hidden unit is linked to a corresponding embedding vector, which is used in the second step to make predictions.

Then, HuBeRT is trained by predicting hidden units in a fashion similar to the wav2vec2 architecture. In essence, raw speech waveforms are first fed to a CNN Encoder which outputs embeddings of the speech signal time frames. Then, some of these embeddings are masked according to a pre-set masking strategy, and fed to a Transformer neural network, whose goal is to infer from the context meaningful information about the masked frames. To do this, the whole model is trained using the cross-entropy loss, where the goal of the Transformer neural network is to accurately predict the hidden units associated to the masked frames, where the prediction of which hidden unit is predicted is done by computing the cosine similarity between the output

of the Transformer model and the embeddings associated to the hidden units. Importantly, one can notice that HuBeRT uses a less complex loss function than wav2vec2.

In summary, HuBeRT differs from wav2vec2 in three main fashions. First, it builds targets via a separate clustering process (*e.g.* using the K-Means algorithm) while wav2vec2 relied on a quantization process using Gumbel-softmax (*i.e.* the combination of a Gumbel prior distribution for quantization and a softmax). Then, the existence of a bank of pseudo-targets allows using the cross-entropy loss rather than a combination of a contrastive loss and a diversity loss, whose behaviour is both better understood by the machine and deep learning literature and requires fewer choices (*e.g.* no decision is needed about fixing a number of distractors to sample). Finally, HuBeRT adaptatively improves pseudo-targets using the embeddings of the convolutional encoder, while wav2vec2 only used quantization in the training process.

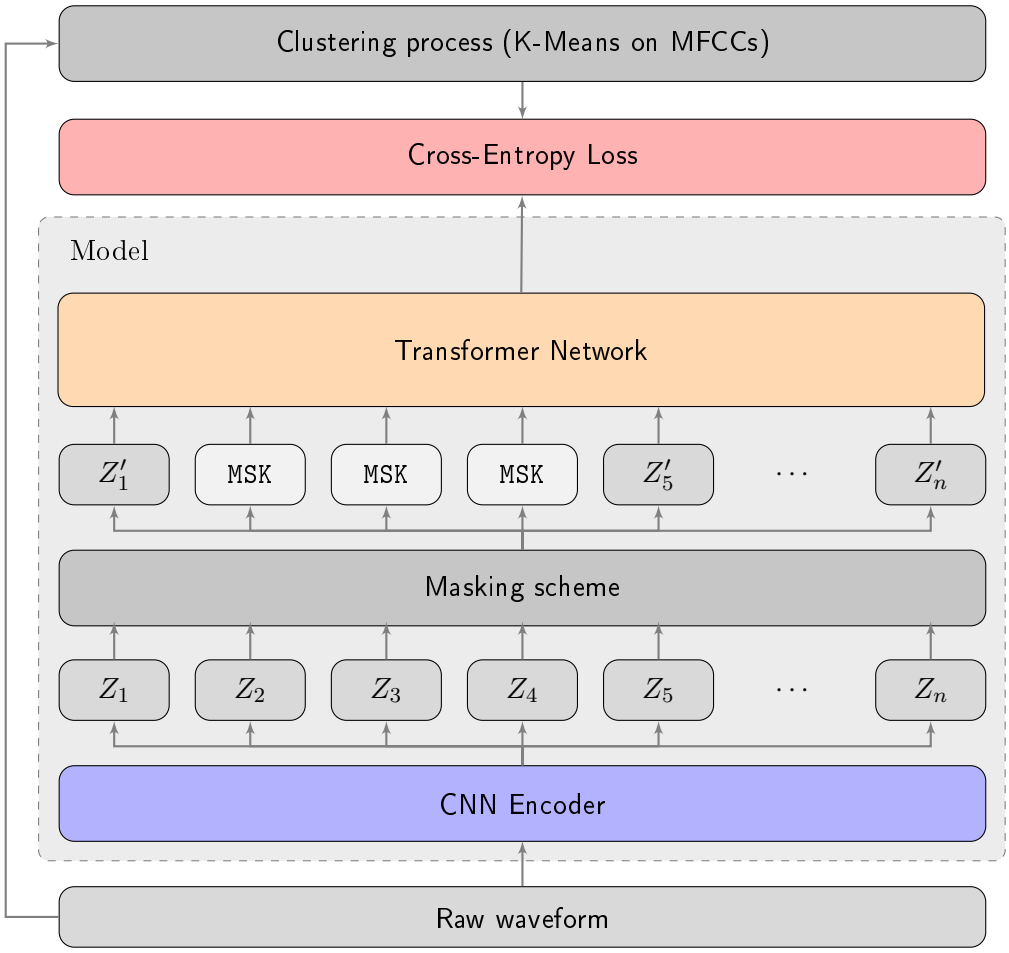


Figure 4: HuBeRT pre-training model architecture

With these improvements, testing in the same configuration as in the wav2vec2 experiments, the HuBeRT model achieves a WER on the clean test subset of LibriSpeech of 1.8 and of 2.9 on the other test subset, slightly improving over the state-of-the-art result attained by wav2vec2, albeit with more parameters (the Large version of the wav2vec2 model contained about 317

million parameters while the X-Large version of the HuBeRT model contains about 2 billion). With roughly the same number of parameters, wav2vec2 and HuBeRT achieve comparable results in terms of WER on the LibriSpeech test set.

WavLM The WavLM model of (S. Chen et al. 2021) is yet another attempt at leveraging self-supervised pre-training for speech processing tasks. However, unlike wav2vec2 or HuBeRT, which are specifically designed for the task of ASR, WavLM targets a wide variety of downstream full-stack speech processing tasks, including speaker identification, speaker verification, speech separation, speaker diarization, and speech recognition.

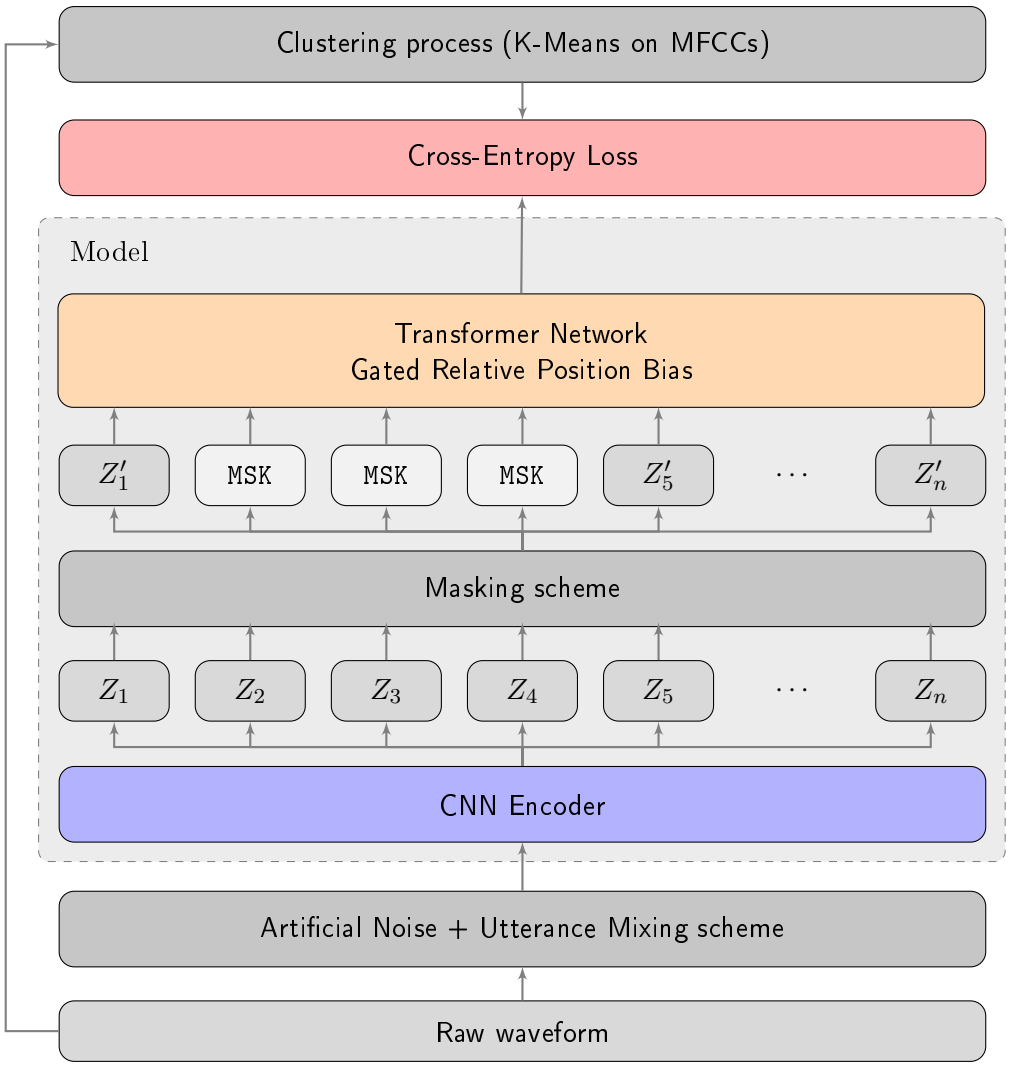


Figure 5: WavLM pre-training model architecture

As displayed in Figure 5, the WavLM model largely inspires itself from the HuBeRT architecture, but adapts it in two main fashions. First, data augmentation is performed as part of training by manually adding simulated noise and overlapping speech by mixing utterances in the pre-training dataset. Although common in supervised learning, data augmentation needs to

be carefully carried when using self-supervised learning so as to not learn non-meaningful representations with regards to the data. Then, WavLM uses modified Transformer blocks, which integrate a Gated Relative Position Bias following Z. Chi et al. (2021), and which allow to better capture the sequence ordering of input speech data.

More importantly, WavLM is built with the underlying idea of generating contextual latent representations of speech that are meaningful across a wide variety of downstream tasks. In this regard, WavLM achieves state-of-the-art performances over multiple tasks evaluated by the SUPERB benchmark (S.-w. Yang et al. 2021). On ASR specifically, when pre-training on a mix of 94,000 hours of unlabelled speech (including the Libri-light corpus, along with 10,000 hours from the GigaSpeech corpus and 24,000 of the VoxPopuli corpus), and fine-tuning on the whole 960 hours of labelled data from LibriSpeech, WavLM achieves a WER of 1.8 on the clean test data subset and of 3.2 on the other test data subset, improving over wav2vec2 by a small margin while keeping the total number of parameters of the model roughly at the same level.

Overall, end-to-end systems have radically changed the performance of ASR systems these last few years. Among all improvements, two main pathways can be set apart. First, the design of models taking raw speech signal as input and which do not rely on a complex chaining of models for speech processing has largely contributed to improvements in the accuracy of ASR systems. In terms of performance on most widely used datasets, one can say with little doubt that end-to-end architectures have largely contributed to filling the gaps left by using multiple bricks in the acoustic modelling pipeline. Then, the use of self-supervised pre-training methods, like in the wav2vec2 or WavLM models, have allowed using speech data in much greater quantity and of much higher diversity when training learning-based ASR systems, resulting in impressive performance gains on traditional evaluation corpora.

Table 1 displays some of the results obtained by the models presented above on the development and testing data subsets of the LibriSpeech corpus. All models were pretrained using either the full Libri-light corpus, or the full Libri-light corpus along with more unlabelled data, and were fine-tuned on the 960 hours of labelled speech data from the LibriSpeech corpus. As can be seen in the results, these models attain comparable performances, with some strong dependence on the language model used in decoding.

Model	Language model	dev		test	
		clean	other	clean	other
wav2vec 2.0 Large	None	2.1	4.5	2.2	4.5
wav2vec 2.0 Large	4-gram	1.4	3.5	2.0	3.6
wav2vec 2.0 Large	Transformer	1.6	3.0	1.8	3.3
HuBERT Large	Transformer	1.5	3.0	1.9	3.3
HuBERT X-Large	Transformer	1.5	2.5	1.8	2.9
WavLM Large	Transformer	NA	NA	1.8	3.2

Table 1: WER of recent end-to-end models on the LibriSpeech corpus

Yet, there remains many open questions in ASR that current systems fail to address completely. As a matter of fact, all results presented above are focused on corpora in English, mostly from audiobooks recordings, hence do not adress multi-lingual and domain adaptation questions.

2.4 Low-resource languages and multi-lingual models

As has been seen, recent deep neural-networks based systems have achieved impressive, human-level accuracy in terms of UER (WER or PER) on ASR in English and on wide-band speech data (*e.g.* on audiobooks recordings). One way forward in research on automatic speech recognition is to try to generalise these results to other languages. Trying to reproduce such results across languages raises two main issues. First, end-to-end models in English such as wav2vec2 (Baevski et al. 2020) are pre-trained on huge amounts of unlabelled data, and fine-tuned on, albeit smaller, still large amounts of labelled data. For many if not all languages, publicly available, clean data, whether labelled or unlabelled, is simply not available in such quantities, especially for languages with few speakers or with languages from countries with little access to technology. Then, models like wav2vec2 were designed specifically for doing ASR in English, and there are no theoretical justifications that models with similar architectures would work well on languages with different phonetics and linguistic properties. However, as assessed by P. Joshi et al. (2020) in a study of the diversity of languages and their respective resources in NLP, more than 90% of the world's population speaks languages that are low resource regarding text data, a figure that is almost undoubtedly higher for speech data resources.

Thus far, the main technique that has been used to tackle the issue of ASR on low-resource languages has been cross-lingual transfers, where learning-based models (HMMs or ANNs) initially designed for one, or more high-resource languages like English, French or Mandarin are adapted in order to be used on low-resource languages. Although there are no theoretical guarantees, the intuition behind cross-lingual transfers is that low-resource languages, although different from their high-resource counterparts, will benefit from the shared language speech features already learned by the model and thus render the learning process faster. Although various teams have shown impressive results by applying cross-lingual transfers for ASR, multiple questions are left out of the spotlight. We detail these results and the questions they raise in this section. In order to better understand these issues, we distinguish three main approaches for training ASR models on low-resource languages, in which fall most of the research on the question.

Language-specific learning A first approach to ASR on low-resource languages is to train a model on the low-resource language data, with no additional data from other languages. Naturally, this approach remains highly limited with regards to the amount of data available to train the model on. Hence, language-specific learning is rarely applicable for low-resource languages if the model used requires large amounts of labelled and/or unlabelled data to be trained, as is the case of modern end-to-end self-supervised architectures like wav2vec2 or WavLM. In fact, self-supervised methods depend crucially on the learning of meaningful contextual representations in the pre-training phase, which require large amounts of unlabelled data. Yet, less data-demanding models and methods can be applied. Recently, however, multiple works and surveys regardings multilingual speech models have highlighted that cross-lingual learning systematically outperformed language-specific learning approaches in ASR experiments (Yadav et al. 2022).

Cross-lingual learning A second approach aims at leveraging the recent highly performant models trained on high-resource languages (particularly in English) and to adapt them to ASR tasks on low-resource languages via a carefully designed training procedure. In more generic machine-learning terms, this corresponds to the setting of transfer learning. In short, with this

approach, a model is typically trained using a high-resource language, and then fine-tuned using the available data in the low-resource language. On models that use only labelled data, this can be done by fine-tuning the model with the low-resource language data in the same way as it was trained using the high-resource language data. In particular, this allows re-using already trained and publicly available models. On models that use both labelled and unlabelled data in a self-supervised way, this approach can differ between either performing only the pre-training phase on the high-resource language data, and then the fine-tuning phase on the low-resource language data, or training the entire model (both phases) using the high-resource language data and fine-tuning the model on the low-resource language data afterwards.

Multi-lingual learning The third and last approach is very similar to the second, but changes from it in the fact that the initial training of the model is not carried on a single high-resource language, but rather multiple high-resource languages or a diversity of languages including both low and high-resource languages. This approach is particularly applied on end-to-end self-supervised methods, where the model is pre-trained on a variety of languages and the fine-tuned on a single target languae. The intuition behind this approach is that training the model on speech from multiple languages will help in learning universal speech representations that can be used across many different languages, and easily fine-tuned to a target language. Additionally, this approach takes advantage of the fact that a single multi-lingual model has to be learned, and fine-tuning can be performed separately on individual languages without the need to retrain the original model. On the other hand, there is no guarantee that universal features can be learned from speech in such a diverse set of languages, nor that, if such universal representations can be learned, that they will be helpful in performing ASR tasks accurately.

Multiple research works have tried to evaluate the performances of each approach in a comparative manner. Thus far, most results tend to indicate that multi-lingual learning yields the best performance for ASR on most low-resource languages (Yadav et al. 2022). In this vein, one seminal work is the approach of Heigold et al. (2013) who perform multiple experiments on 11 languages, comparing mono-lingual DNN-HMM systems to cross-lingual and multi-lingual systems. In this study, using only labelled data, multi-lingual models performed better than cross-lingual and mono-lingual approaches on all considered languages. On systems using both unlabelled and labelled data, Riviere et al. (2020) show that unsupervised pre-training transfers well from English to multiple low-resource languages,leading to results on par or outperforming models trained solely via supervised training on the low-resource language data, showing that cross-lingual learning surpasses language-specific learning in most settings. In that same vein, but extending the approach to multi-lingual learning, (Conneau et al. 2020) study the effects of training unsupervised multi-lingual representations using the wav2vec2 architecture, and the trade-offs associated to using larger speech corpora from multiple languages. In a first experiment, the authors show that pre-training the model on English only outperforms mono-lingual models, confirming the results of Riviere et al. (2020). Then, in a second experiment, multi-lingual pre-training is applied using unlabelled data from 10 languages before the same fine-tuning procedure that in the first experiment is applied. Using this experimental protocol, the authors show that pre-training on multiple languages improves the accuracy of the ASR system across all tested languages and for all metrics than pre-training on a different language, and often outperforms models pre-trained and fine-tuned on data from the low-resource languages.

Currently, the state-of-the-art in multi-lingual self-supervised models is the XLS-R model of Babu et al. (2021). XLS-R consists in a model with the same architecture as wav2vec2, although with more parameters and aimed at multiple downstream tasks (automatic speech translation, automatic speech recognition, language identification and speaker identification). It is pre-trained on speech audio from 128 languages, with the end-goal of making a model with generic, discriminative features available for fine-tuning on target languages (not necessarily in the 128 languages). To this end, Babu et al. (2021) pre-train XLS-R on 436,000 hours of data, of which 372 hours come from the VoxPopuli corpus, 50,000 hours from the multilingual LibriSpeech corpus, 7,000 hours from the CommonVoice crowdsourcing project, 6,600 hours from the VoxLingua corpus, and 1,000 hours from the BABEL corpus. Data for pre-training include both high and low-resource languages. To account for the extreme differences between high-resource languages such as English and much lower-resource languages (*e.g.* Sinhala, Zulu, etc.), training batches are sampled to contain data from multiple languages using a sampling distribution with an upscaling factor, which controls the trade-offs between high and low-resource languages in training. For the task of ASR, XLS-R is evaluated using: the BABEL ASR development set, the Multi-lingual LibriSpeech dataset with a selected split for evaluation, the CommonVoice dataset with selected subsets, the VoxPopuli test sets, and the LibriSpeech development and testing sets.

On the CommonVoice corpus, using one hour of labelled data per language for fine-tuning and without language modle, XLS-R outperforms all previous state-of-the-art models (*e.g.* m-CPC of Riviere et al. (2020) or the previous iteration of XLS-R trained on 53 languages) in terms of PER. The large version of the XLS-R model obtain PER of 2.0 on Spanish, 3.9 on French, 3.5 on Italian, 4.1 on Kyrgyz, 4.2 on Dutch, 4.1 on Russian, 5.5 on Swedish, 4.4 on Turkish, 3.4 on Tatar and 15.7 on Chinese from Hong-Kong. On the multi-lingual LibriSpeech corpus, fine-tuning on 10 hours of speech and using the included language models, WLS-R achieves a WER of 10.9 on average across 8 languages: 12.9 on English, 7.4 on German, 11.6 on Dutch, 10.2 on French, 7.1 on Spanish, 12.0 on Italian, 15.8 on Portuguese and 10.5 on Polish. Finally, on the original LibriSpeech corpus, the large version of XLS-R attains results close to the wav2vec2 model pre-trained on English only, with on average a 0.7 points of WER difference.

2.5 Domain adaptation for ASR models

As mentionned above, a considerable difficulty in constructing adaptable end-to-end ASR systems is the presence of a domain shift between the data on which the model is trained and the data on which it will be used in practice. Current state-of-the-art offline end-to-end ASR models in English have been shown to match or improve over human performance on multiple instances in settings where the training and evaluation datasets are close in terms of content and prosody. However, to the best of our knowledge, all recent architectures fail to keep pace when the evaluation dataset differs extensively from the training data. In this section, we briefly outline the task of domain adaptation and its typical applications and methods, before presenting some recent results in domain adaptation for ASR.

As highlighted in Section 2.3, in English, large amounts of speech data are already publicly available in audiobooks corpora such as Librispeech or via crowdsourcing projects like CommonVoice. However, it is reasonable to expect that many real-life conditions for using ASR systems differ broadly from the studio conditions of audiobooks recording, or even from the validated,

high-quality, isolated speech data gathered in crowdsourcing projects. In fact, one may wish to use ASR systems in settings with much more background noise such as when using phone voice assistants or when transcribing TV interviews, as well as with offline data from sources using different frequency ranges like speech data from telephone or radio conversations. However, as of today and to the best of our knowledge, little labelled data is publicly available from such sources, rendering training adaptable end-to-end ASR models arduous. In theoretical terms, this difference between the training data, which originates from a source domain (*e.g.* an audiobook corpus) and the data onto which the system is to be used, which belongs to another, different but related domain (the target domain) is called a domain shift.

Multiple works have shown that a domain shift between source and target domains almost systematically induces a sharp decrease in performance in speech recognition tasks. Notably, Hsu, Sriram, et al. (2021) highlight using multiple commonly used datasets from varied domains that, if not addressed, differences in the source and target domain data in the wav2vec 2.0 model lead to critical performance gaps. In particular, a wav2vec 2.0 base model pre-trained on the 452 hours of unlabelled speech data from TED conferences of the TED-LIUM v3 dataset and fine-tuned on a 10 hours subset of labelled data from that same corpus achieves a WER on the Librispeech `dev-other` dataset of 23.4%, while pre-training and fine-tuning directly on the training sets of the Librispeech corpus yields a WER of 10.53% (+123% relative increase). Conversely, a wav2vec 2.0 model pretrained on the 960 hours unlabelled data from Librispeech, and fine-tuned on 10h of labelled data of the same corpus achieves a WER of 12.92% on the TED-LIUM v3 `dev` dataset, while pre-training and fine-tuning directly on the training sets of the TED-LIUM V3 corpus yields a WER of 9.93% (+30% relative increase).

Moreover, these limitations in the generalization capabilities of ASR systems are not unique to self-supervised end-to-end models, and the same pattern of performance gap can be observed for fully supervised approaches. Using an ASR Conformer model trained on the labelled Librispeech dataset, Sukhadia et al. (2022) show that even with a mild domain-shift, ASR evaluation metrics can degrade significantly. Using the WSJ dataset consisting of clean speech data of read news articles from the Wall Street Journal as test set, the authors observe a 241% relative mean increase in WER on the `test` and `eval` subsets, highlighting that naive vanilla transfer learning can fail even when domains are close. In a similar fashion, Hwang, Misra, et al. (2021) use data from video-captioning on YouTube and Google Home voice commands as a source domain and data from natural phone conversations from call centers as a target domain. When training a fully-supervised Conformer RNN Transducer on the source domain and testing it on the target domain, their approach achieves a WER of 6.2%, while training the same model directly on data from the target domain yields a WER of 3.2% (+93% relative increase).

In broad terms, attempts to correct or mitigate the degradation in performance due to a change of domains can be grouped under the name of domain adaptation. More specifically, domain adaptation is a transductive type of transfer learning which aims at making it possible to use a learnable algorithm trained on data from a source domain on a different but similar target domain. Multiple techniques have been experimented for domain adaptation in the case of ASR, particularly with the recent development of end-to-end ASR models. For the sake of clarity with respect to the task at hand, we mention in this section all approaches that aim at making the end model adapted to the target domain data distribution, even if these do not specifically match the definition of domain adaptation and do not involve a specific learning-based step.

Pre-processing adaptations Pre-processing adaptations encapsulate all data-centric methods that aim at applying transforms to the source domain data in a way such that it matches or resembles the target domain data distribution. These transforms typically leverage some expert prior knowledge about the data at hand, and can be carried with or without the help of a learning algorithm. As such, the domain adaptation is performed as a pre-processing step, prior to training a learning model. In other words, the domain adaptation is performed as a separate step, outside the model and before training, so that the transformed source domain data can directly be used as surrogate to train the end model in a way that is directly adapted to the target domain. If the source domain is an audiobook corpus, and the target domain a corpus of telephone conversations, one example of pre-processing adaptation would be to simulate passing through a phone line for all audio inputs from the source domain, and then training on the resulting data. There exists a vast literature on such adaptations for specific cases, such as the emulation of phone conversation from clean speech, but to the best of our knowledge, few examples of this type of adaptation are present in the academic literature on ASR. Recently, however, R. Joshi et al. (2022) use a single speaker text-to-speech engine to simulate audio data from a target domain in order to obtain a baseline for domain adaptation methods in ASR.

Reweighting Reweighting algorithms are also data-centric methods, whose aim resides this time in producing resampling weights for the source domain data in order to match as closely as possible the distributions between the source and target domains data. In that sense, reweighting algorithms differ from out-of-model adaptations by the fact that they do not alter the source domain data based on the target domain data distribution estimate, but rather weight the available data points in the source domain so as to match the distribution of the target domain without estimating it directly. Naturally, reweithing is only feasible when the source and target domains differ only by a small margin, such that the distribution of the data in the source domain contains samples that, to a margin, fully represent the diversity of samples in the source domain. Reweighting is thus particularly useful in cases where the target and source domains have the same definition domain in mathematical terms, or overlap to a large extent.

Pseudo-labelling Pseudo-labelling is an iterative domain adaptation method which aims at using the source domain data to train a first iteration of a learning algorithm, and to use this model to label data from a target domain. Then, once the target domain data points are labelled, the model is either trained again using a combination of the source and target domains, or fine-tuned on the target domain. In practice, as highlighted on computer vision by Saito et al. (2019), it is often desirable for increased performance to have a few labelled data points from the target domain during the first phase of the training procedure, even if these points constitute a negligible proportion of the training data.

Domain-adversarial training Domain adversarial training is a learning-based domain adaptation method introduced by Ganin, Ustinova, et al. (2016) that falls in the category of representation learning methods, and that aims at rendering the latent representations of the data given by a model inefficient to discriminate between the training source data and the target data using a complimentary neural network. In short, domain adversarial training is usually carried by training a model with a combination of losses, in which one loss helps in solving the end task of the model, and another loss helps in making the latent representations of the data in the

model insensitive to domain shift. In practice, this procedure is usually carried by plugging a secondary model, called the adversary, on the latent representations of the first model, and to train it to discriminate between data from the source and target domain, and update the weights of the main model using a gradient reversal layer (so as to train the model in a way such that the latent representations are unhelpful in discriminating whether the data originates from the source or target domain). Although well adapted to tasks where representation learning plays a key role in the studied downstream tasks, domain adversarial training is notably difficult to use in practice due to unclear heuristics about which weights need be updated using the reversed loss of the arbitrary, and on the trade-offs at hand when making the system unable to model domain membership.

Post-processing adaptations Post-processing domain adaptation methods are methods that try to overcome the problem of domain shift via post-processing procedures, *i.e.* after the model has been trained on the source domain and without using a learning algorithm. In the case of ASR, multiple works have explored this approach by trying to use a language model tuned to a target domain in the decoding phase of a system using a self-supervised model trained on a source domain (Zhou et al. 2021; Z. Meng et al. 2021).

In comparison with the research on multi-lingual ASR, to our knowledge, little work has been done to analyze the advantages and drawbacks of one specific domain adaptation method over the other, and most research has been carried with the idea of testing a single approach at a time. In our experiments, we focus exclusively on pre-processing adaptations, essentially by experimenting with self-supervised models pre-trained on a source domain and fine-tuned on a target domain.

2.6 Other unresolved tasks and open questions in ASR

Multi-lingual models and domain adaptation for speech signal processing are two important limitations to most of the approaches in ASR, and constitute two open and active research questions. Yet, several other limitations exist, some of which we do not explore in this report. A first difficulty in the task of processing speech is the possibility of overlapping spoken language in recordings. In this regard, a wide array of research endeavours focus on the task of speech separation, which aims at separating spoken language from multiple speakers in the same audio recording. In this regard, the recent work of Zeghidour et al. (2021) offers a concise overview of the variety of models used by the literature to tackle speech separation.

Likewise, an inherent difficulty lies in isolating speech content in audio recordings that do not contain spoken language only (*e.g.* interviews, phone conversations, etc.), and in isolating a individual speaker in a conversational recording. The field of speaker identification focuses on this particular issue. In this regard, Sidorov et al. (2013) give a detailed account of the different methods that allow overcoming this issue. Finally, despite the increased availability of data, both labelled and non-labelled, multiple challenges are still faced regarding data when training ASR models: variety of speakers, tones, pronounciations, accents, etc. In this regard, multiple works have tried to put forward approaches for data augmentation in speech processing, including towards the task of ASR, notably in conjunction with recent architectures, like in the WavLM model pre-processing pipeline (S. Chen et al. 2021).

3 Resources and experimental setup

As highlighted in the previous sections, training self-supervised end-to-end ASR models requires large amounts of both unlabelled and labelled data, as well as considerable computation resources. In addition, all such models have been shown to be highly dependent to a wide variety of hyper-parameters and, in broader terms, to their experimental setup. In this section, we briefly present the datasets and languages commonly used when training and evaluating ASR systems, and often mentionned by the literature on the subject, then give an overview of the publicly available datasets on a range of languages, and present the experimental protocol associated with pre-training, fine-tuning and evaluating wav2vec2 and XLS-R models.

3.1 Data and languages

Despite the recent advances made towards using unlabelled data as part of training end-to-end ASR systems, datasets have remained essential assets in researh on this topic. This subsection gives a short overview of three commonly used datasets used in training and testing ASR systems, along with their features such as the availability of transcriptions, the languages spoken or the amount of speech data.

LibriSpeech (LS) LibriSpeech (Panayotov et al. 2015) is a widely used corpus for speech-to-text tasks, which is part of the wider LibriVox project (see below) and which consists of about 1000 hours of speech from audiobooks recordings sampled at 16kHz along with their transcriptions. Data subsets in the LibriSpeech corpus are: a training set of roughly 960 hours, two development sets of roughly 5 hours, and two testing sets of roughly 5 hours as well. The development and testing subsets are split into: a subset designated as clean, consisting of excerpts read by speakers with low word-error rate (WER; and another subset designated as other, consisting of speech from speakers with higher WER. Additionally, to account for low-resource settings, the training set is typically divided in subsets of respectively 10 minutes, 1 hour, 10 hours and 100 hours of speech. In line with the corpus' source, all speech is spoken in English by professional readers.

Libri-light (LL) Libri-light (Kahn, Rivière, et al. 2019) is another corpus of spoken English audio files derived from the LibriVox project and designed at providing data for training ASR systems with limited or no supervision. It consists of 60,000 hours of audio from audiobooks recordings sampled at 16kHz. Since it originates from the LibriVox project, and consist of audio data with no transcriptions, Libri-light typically uses the LibriSpeech development and testing datasets as benchmark. Alike LibriSpeech, all speech is spoken in English by professional readers.

Common Voice (CV) The CommonVoice project, run by Mozilla, is an open-source, crowd-sourced speech data gathering platform where users are asked to read out sentences in their native languages, or to assess the quality and validity of recordings made by other users. In its ninth release, CommonVoice contains about 15000 hours of recorded speech sampled at 16kHz with corresponding transcriptions in 93 languages. Data subsets are not all split in training, development and testing sets, hence altering the potential for reproducibility in publications.

Speech is recorded in a wide array of languages, and with varying recording conditions although almost always in good quality.

In the following, we mention numerous works that base themselves on these datasets for training or evaluation, and will base our training and decoding procedures solely on them (with the exception of language models trained on external text-only corpora). Likewise, we focus on only two languages: English due to the wide availability of training and testing resources, along with French. However, due to data availability and to the variety of different objectives in research on ASR, a wide array of research endeavours make use of other speech datasets, and often do not use LibriSpeech, Libri-light or Common Voice as a baseline. Hence, for the sake of clarity, we cover a large number of other datasets typically used in ASR research in the appendix.

3.2 Training procedure and modules

As highlighted in Section 2, training of a fully end-to-end self-supervised model contains multiple steps which all involve a learning phase, and which are inter-dependent towards the goal of achieving good transcription of speech data. The interweaving of these components make training a fully end-to-end, self-supervised ASR model particularly arduous, all the more given that evaluating the quality of the model can hardly be done in . In the following section, we go into further detail on each of these steps for the training of a wav2vec2 or XLS-R model. In particular, after reviewing the training steps, with detail our understanding of the influence of hyper-parameters during training, as well as give intuitions regarding the efficient training of the whole model.

Pre-processing

As with most deep-learning based systems, pre-processing of the data at hand plays a crucial role in the efficacy of the model being trained for the task considered. In this regard, pre-processing of speech data is and arduous task for two main reasons: first, while obtaining open-source audio data is relatively easy, isolating speech from other audio components is an active research question itself, hence constructing large corpora composed of speech data only remains difficult; then, data used in speech recognition systems must necessarily contain some labelled data, which imply not only gathering transcriptions but also processing so as to make the output uniform, hence making choices with are not neutral and impact the quality of the ASR system. In the following, we mostly ignore these questions and consider as a given the unlabelled and labelled datasets used for training the model. In particular, as highlighted in the presentation of the datasets, the speech audio excerpts used in training contain only isolated speakers, and are well-segmented so as not to include long periods of silence.

On top of this, two pre-processing choices need to be made, which again typically have a decisive impact on the performance of the model. First, regarding audio data, a choice of whether to feed the model the raw audio digitally-encoded waveform, or some transformation of this waveform, needs to be made. In this report, the models that we rely on experimentally all use the raw waveform as input. Nevertheless, for completeness, one should mention that multiple acoustical representations exist and have often been used as input for deep-learning based ASR systems. These typically represent an audio waveform locally (so as to preserve the non-stationnary property of speech signals), in the frequency domain to render interpretations of

the signal easier, and with features of relatively low dimension so as to reduce the computational cost of dealing with the input. Examples of transformations that are used as input to ASR systems are the spectrograms (representation of the spectrum of frequencies of a signal as it varies with time) or mel-frequency cepstral coefficients (representation of the short-term power spectrum of a signal).

The second pre-processing choice that needs to be made concerns the textual data contained in transcriptions. As a matter of fact, in order for a model such as wav2vec2 or XLS-R to output logits corresponding to the relative probabilities of a speech segment to correspond to a grapheme, one needs to set in marble beforehand the dictionary of graphemes that may be predicted. This step is known as tokenization. In the experimental part of this report, tokenization will always be done by choosing characters, where the list of characters corresponds to a common list of characters in the considered language (*i.e.* the alphabet letters along with some variations like accents, and a few punctuation marks). However, for completeness, it is worth noting that many other tokenization approaches exist. In particular, some recent approaches in ASR are using word-pieces instead of characters, where the dictionary of graphemes consists in a large list of the most frequent word-pieces observed in a given corpus (*e.g.* `BLE`, `AT`, `DO`, etc.). Although, on the one hand, this approach is more sensible in the sense that word-pieces always correspond to one or a few pronounciations (unlike letters which are sometimes silent), on the other, its major drawback is that it fixes a reduced set of word-pieces and diminishes the ability of the model to produce out-of-vocabulary (OOV) output. Finally, other approaches benefiting from well-annotated datasets rely on phonemes rather than graphemes, which better fit the aim of the model to transcript speech but which also hinder its practicality as the output of the model is therefore a list of phonemes, which is, to most people, harder to read than a sentence written in the usual alphabet characters.

Pre-training

The pre-training phase consists in using non-labelled speech data (*e.g.* in the form of recordings with identified segments of speech they contain) to help the model learn meaningful representations of the input data without ever using transcriptions. Compared to fully-supervised approaches, which can theoretically be trained in the same way with only this step skipped, the purpose of this first phase is specifically to leverage the knowledge that can be inferred from speech data alone, so that this knowledge can be passed-on via the model parameters, and used in supervised learning for some downstream task, in this case ASR.

For convenience and readability, we display the architecture of the wav2vec2 model again in Figure 6 below. Moreover, we define the problem in a more mathematical-heavy formulation, which should help in clarity. In that regard, let $T_0 \in \mathbb{N}$, and $\mathbf{X} = (x_1, ..., x_{T_0})$ be an audio recording of size T_0 with $\forall i \in \{1, ..., T_0\} : x_i \in \mathbb{R}$. In other words, the audio recording is a T_0-dimensional vector, where T_0 is fixed for a single recording but potentially varies between different recordings. For instance, suppose that we consider a clip of speech of 15.31 seconds, encoded at a sample rate of 16kHz. Then, the total clip can be seen as a real vector of size $15.31 \times 16000 = 244,960$. For simplicity, we consider that this vector takes values in $\mathbb{R}$ although it technically takes values in a given range determined by the bit depth and the codec used to store the audio signal digitally.

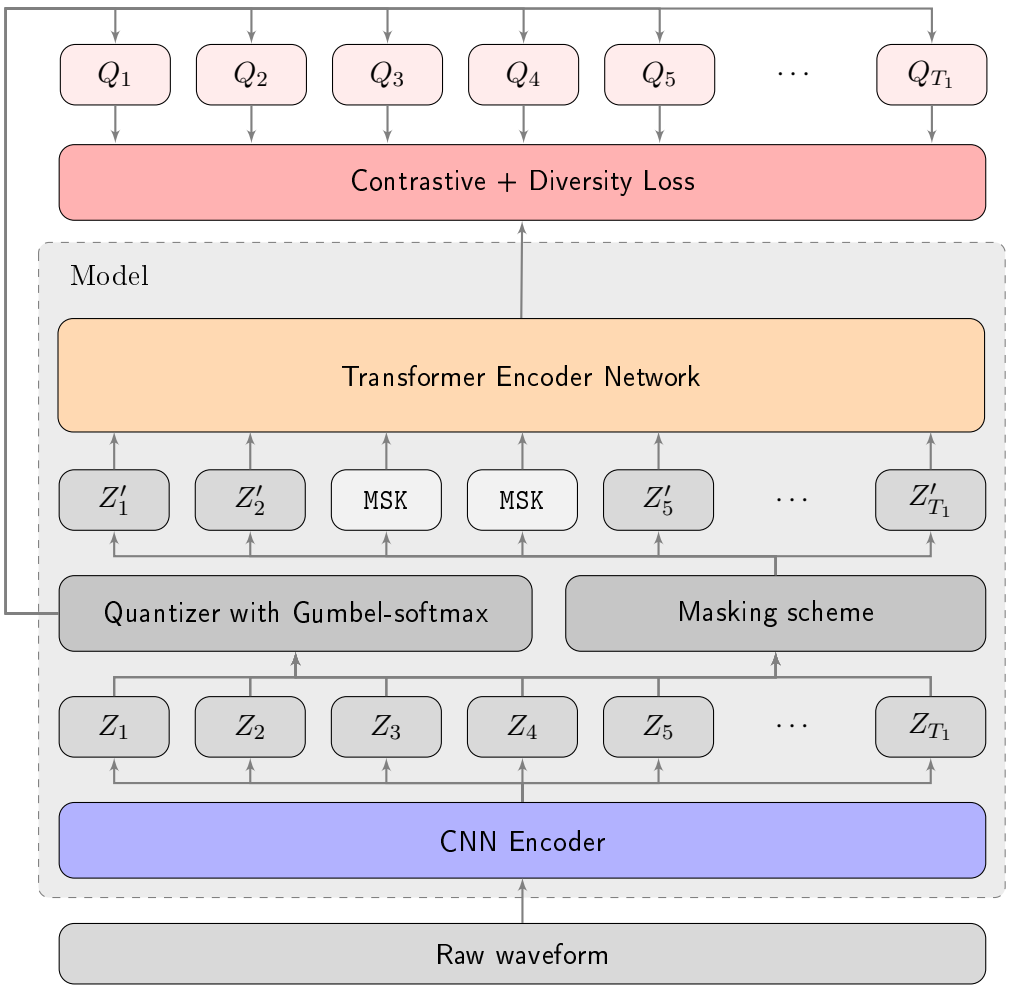


Figure 6: wav2ec2 pre-training model architecture

As displayed, the first module of the pre-training architecture is a CNN Encoder. Its role is both to reduce the dimensionality of the audio data and transform it into features that take into account local dependencies in the sequence. In short, it converts the raw audio waveform into a sequence of feature vectors $\mathbf{Z} = (Z_1, ..., Z_{T_1})$ with $\forall i \in \{1, ..., T_1\} : Z_i \in \mathbb{R}^{M_1}$, where, in practice, $M_1 = 512$ for the wav2vec2 and XLS-R models. In other words, the CNN Encoder takes a digitally encoded raw audio waveform of a given size T_0 as input, and maps it of a sequence of $T-1$ latent feature vectors, each of dimension 512. The internal architecture of this CNN Encoder is composed of: one layer of normalization that sets the raw audio waveform to have zero mean and unit variance, followed by 7 one-dimensional convolutional layers, including layer normalization and with the GeLU activate function, where, denoting as Φ the cumulative distribution function of a standard gaussian probability distribution, GeLU is defined as:

$$\text{GeLU} : \begin{cases} \mathbb{R} \to \mathbb{R} \\ x \mapsto \text{GeLU}(x) = x\Phi(x) \end{cases}$$

For instance, suppose that, the input of the neural network is a 15.31 seconds clip of speech. With a sample rate of 16kHz, this implies that the input vector is a vector of size $15.31 \times 16,000 = 244,960$, hence, using the notations above, we have $T_0 = 244,960$. In the CNN Encoder, this vector is first set to have mean zero, and unit variance. Following this, the vector goes through the 7 convolutional layers. These layers all have 512 output channels. Hence, the 244,960-long input vector is, at each pass through a convolutional layer, reduced to latent feature vectors with 512 channels, where the number of these vectors at each pass depends on the parameters of the convolutional layers, *i.e.* the kernel size, stride, padding, and dilation.

In practice, the 7 convolutional layers have, respectively: kernel sizes 10, 3, 3, 3, 3, 2, 2; stride 5, 2, 2, 2, 2, 2, 2; padding set to 0 and dilation to 1. The number of latent feature vectors at each pass can be computed by the following formula, where T_{in} corresponds to the number of latent feature vectors at input, T_{out} to the number of latent feature vectors at the output of the layer, and kernel_size, stride and dilation the respective paremeters:

$$T_{\text{out}} = \lfloor \frac{T_{\text{in}} + 2 \times \text{padding} - \text{dilation} \times (\text{kernel\_size} - 1) - 1}{\text{stride}} + 1 \rfloor$$

Omitting calculations, the convolution layers yield the following number of latent feature vectors: after the first layer, 48,991 vectors of size 512; after the second layer, 24495 vectors of size 512; after the third layer, 12247 vectors of size 512; after the fourth layer, 6123 vectors of size 512; after the fifth layer, 3061 vectors of size 512; after the sixth layer, 1530 vectors of size 512; and after the seventh and last layer, 765 vectors of size 512. Hence, the CNN Encoder maps the 15.31 seconds clip of speech into 765 latent feature vectors, each of them having 512 channels.

To put things in perspective, this implies that each of the 765 vectors corresponds to roughly $15.31/765 = 0.02$ seconds of audio, *i.e.* 20 miliseconds. Hence, at the output of the CNN Encoder, the frequency, measured by the number of latent feature vectors per second of audio, is at 49Hz. Finally, given the kernel sizes and strides, one can compute the receptive field of the CNN Encoder as follows, denoting by r the receptive field, and indexing the kernel sizes and strides by the layer:

$$r = \sum_{l=1}^{7} \left(\left((\text{kernel\_size}_l - 1) \times \prod_{k=1}^{l-1} \text{stride}_k \right) + 1 \right) = 401$$

In other words, the receptive field of the CNN Encoder is of roughly 401 data points in the original digitally stored audio file, which, at 16kHz, corresponds to roughly 25 miliseconds. In other words, the latent feature vectors that are output by the CNN Encoder roughly encode 20 miliseconds of speech each, as well as take into account local dependencies in a 25 miliseconds window around the considered data sampled now at 50kHz.

Then, once the raw audio waveform has been passed through the CNN Encoder, two copies of the T_1 latent feature vectors are created. In the rest of this section, we will refer to these two copies as the unmasked, and input copies, the first one consisting in the features that will be used as the ground truth that the Transformer network learns to predict, and the second being the one that will be partly masked and fed to the Transformer network. In order to fit the dimension of the output vectors of the Transformer network, the input feature are fed to a dense

linear layer so that the latent feature vectors have dimension 768 for the BASE model, and 1024 for the LARGE model, rather than 512. In the following, we will denote these input latent feature vectors by $\mathbf{Z}' = (Z'_1, ..., Z'_{T_1})$ with $\forall i \in \{1, ..., T_1\} : Z'_i \in \mathbb{R}^{M_2}$ and $M_2 = 768$ or $M_2 = 1,024$ in practice. The unmasked latent feature vectors, however, are ketp to have dimension $M_1 = 512$. Then, the unmasked features follow a quantization procedure, and the input features are passed through a masking scheme before being input to the Transformer network.

For the masking scheme, a proportion p of the input latent feature vectors are sampled, where p takes the value of 6.5% in practice. Then, all 10 time-steps following the time-steps selected by sampling are masked (with no regards to whether they were already masked/selected), in the sense that these input latent feature vectors are replaced with a unique mask feature vector, that we will denote by $Z_{\text{M}} \in \mathbb{R}^{M_2}$, and that is a learned parameter of the model. The indices of the masked vector are also stored, so that they can be used to retrieve the masked features at the time of the computation of the contrastive loss. In the following, we denote by $\mathbf{Z}'' = (Z''_1, ..., Z''_{T_1})$ with $\forall i \in \{1, ..., T_1\} : Z''_i \in \mathbb{R}^{M_2}$ these feature vectors which have been partly masked, and which serve as input to the Transformer neural network.

On the other hand, the unmasked latent feature vectors are fed through a quantizer module. As a matter of fact, these unmasked latent feature vectors are continuous in nature (*i.e.* take value in the reals), as they are resulting from convolution operations on the raw audio waveform data, which, in practice, can be seen as a series of real numbers. However, a Transformer neural network (as introduced by Vaswani et al. (2017)) requires a finite, fixed amount of token slots, hence a discrete set of values from which the Transformer network can choose. The role of the quantizer is thus to convert the unmasked latent feature vectors, which are continuous, into discrete vectors, all while retaining as much discriminative information as possible, and keeping differentiability properties so as to be able to backpropagate the gradient from the loss, all the way to the CNN Encoder.

This operation is not straightforward, as mapping continuous features to a discrete space while retaining differentiability properties is a long-standing problem, In the wav2vec2 model, this is solved using an automatic learning of a finite number of speech codewords, grouped in two codebooks, from which speech features are created by sampling from the Gumbel-Softmax distribution, so that the features are made of the speech codewords. In our setting, wav2vec2 and XLS-R both use 640 speech codewords, split in two codebooks of 320 speech codewords, such that, in theory, the model can express up to $320 \times 320 = 102,400$ speech units (each of them made of two speech codewords).

In more details, and with more mathematical abstraction, recall that, following our example, we have T_1 unmasked latent feature vectors of size 512 obtained from the CNN Encoder. Let $\mathbf{Z}$ represent these vectors, such that $\mathbf{Z} \in \mathbb{R}^{T_1 \times 512}$. In the quantizer module, these vectors are multiplied by a quantization matrix $\mathbf{W}_q \in \mathbb{R}^{512 \times 640}$, composed of parameters that are learned by the model during training, to obtain logits which one can think of as the relative probability of the feature vector to correspond to the speech codewords in the codebooks. In other words, the multiplication of the unmasked latent feature vectors with the quantization matrix yields a matrix $\mathbf{L}$ of size $T_1 \times 640$, which can be seen as the horizontal concatenation of two matrices $\mathbf{L}_1$ and $\mathbf{L}_2$, each of size $T_1 \times 320$ and which can be seen as the logits for each latent feature vector over the speech codewords from the two codebooks.

Then, samples from a categorical distribution are drawn using as class probabilities the logits from the matrix $\mathbf{L}$, using an adaptation of the Gumbel-Max trick introduced by Gumbel (1954) and Maddison et al. (2014) known as the Gumbel-softmax (Jang et al. 2016), which, depending on a temperature parameter, approaches a categorical distribution on the logits while retaining differentiability properties, and hence allowing for backpropagation. In short, the Gumbel-softmax allows to draw efficiently samples from a categorical distribution with given class probabilities, in a way such that, as the temperature parameter approaches zero, the sampling becomes asymptotically close to an argmax operator. In other words, this allows sampling with noise in a way that mimics the argmax operator while retaining differentiability, and which adds noise depending on a set hyperparameter and the Gumbel distribution.

This sampling results in a matrix $\mathbf{O} = [\mathbf{O}_1, \mathbf{O}_2]$ such that $O \in \mathbb{R}^{T_1 \times 640}$ and $\mathbf{O}_1, \mathbf{O}_2 \in \mathbb{R}^{T_1 \times 320}$ so that each matrix $\mathbf{O}_1$ and $\mathbf{O}_2$ contains, for each latent feature vector, a one-hot vector of size 320 indicating the index of the speech codeword corresponding to the latent feature vector for the first and second codebooks respectively. These matrices are sparse (with only two non-null values in each row, both equal to one) and allow to select the speech codewords using a quantization projection matrix, which maps the matrix $\mathbf{O}$ containing the indices of the speech codewords to use to a resulting matrix $\mathbf{Q} = [\mathbf{Q}_1, \mathbf{Q}_2]$, such that $Q \in \mathbb{R}^{T_1 \times M_3}$ and $\mathbf{Q}_1, \mathbf{Q}_2 \in \mathbb{R}^{T_1 \times 2M_3}$, where the codewords are learned parameter of size M_3, where, in practice, $M_3 = 128$ for the Base model and 384 for the Large model. In these final matrices $\mathbf{Q}_1$ and $\mathbf{Q}_2$, the rows contain the speech codewords corresponding to the unmasked latent feature vectors. In this quantization scheme, the codewords are learned parameter and, as underpinned below, are made to span as much as possible the underlying latent space using a diversity loss function.

To review the first few steps of the pre-training phase, at this stage of the training pass, the data consists in a first set of latent feature vectors $\mathbf{Z}'' \in \mathbb{R}^{T_1 \times M_2}$ which have been partly masked and which will serve as input to the Transformer neural network, and a second set of unmasked, quantized latent feature vectors $\mathbf{Q} \in \mathbb{R}^{T_1 \times 2M_3}$, which will serve as an artifical ground-truth which the Transformer neural network is trained to retrieve. In practice, for the Base version of wav2vec2, these matrices are such that $\mathbf{Z}'' \in \mathbb{R}^{T_1 \times 768}$ and $\mathbf{Q} \in \mathbb{R}^{T_1 \times 256}$; and, for the Large version, such that $\mathbf{Z}'' \in \mathbb{R}^{T_1 \times 1024}$ and $\mathbf{Q} \in \mathbb{R}^{T_1 \times 768}$.

Then, the core of the wav2vec2 model is its context network, which consist of a modified version of the encoder of a Transformer neural network, and which takes as input the latent feature vectors $\mathbf{Z}''$ and maps them to contextualised representations by passing them through a relative positional embedding layer, followed by M_T Transformer blocks, where M_T is set to 12 for the Base version of the model and to 24 for the Large version. The first part of this context network consists in a grouped convolutional layer taking as input the masked latent feature vectors, which uses multiple convolutions (*i.e.* multiple, potentially different kernels in a single layer) resulting in multiple channel outputs, to learn a varied set of low and high level latent feature vectors which in turn serve as relative positional information to the network. These positional embeddings are added to masked latent feature vectors $\mathbf{Z}''$ before being fed to the Transformer encoder network, in the same manner that the original Transformer network takes for input a sum between the original inputs and pre-generated absolute positional information in the form of embeddings. Similarly to the case of the CNN Encoder, more details on how a Transformer encoder network works are available in the appendix.

The output of the Transformer encoder network consists in a matrix of context vectors $\mathbf{C} = (C_1, ..., C_{T_1})$ with $\forall i \in \{1, ..., T_1\} : C_i \in \mathbb{R}^{M_2}$ such that all latent feature vectors at the input of the context network are mapped to a context vector, which encodes features that are able to model long-term dependencies. Finally, these contextualised representations are fed to a final projection layer whose aim is to match the dimension of the quantized unmasked latent vectors $\mathbf{Q}$, hence mapping the output $\mathbf{C} \in \mathbb{R}^{T_1 \times M_2}$ to a matrix $\mathbf{C}' \in \mathbb{R}^{T_1 \times 2M_3}$ using a simple linear layer. To put things in perspective, for the BASE model, this corresponds to mapping the output $\mathbf{C} \in \mathbb{R}^{T_1 \times 1,024}$ to a matrix $\mathbf{C}' \in \mathbb{R}^{T_1 \times 768}$, and, for the LARGE model, mapping the output $\mathbf{C} \in \mathbb{R}^{T_1 \times 768}$ to a matrix $\mathbf{C}' \in \mathbb{R}^{T_1 \times 256}$.

At this stage of the training phase, all the data needed to for the loss functions have been computed. As a reminder to the reader, the data now consists in: a matrix $\mathbf{C}'$ containing the contextualised representations output by the Transformer encoder, which have been obtained by feeding the masked latent feature vectors to the context network; a matrix $\mathbf{Q}$ containing the quantized unmasked latent feature vectors, and a list of the indices corresponding the to latent feature vectors that were masked. On top of that, for each masked speech time-frame, a number M_D of distractor samples are drawn uniformly from the unmasked quantized latent feature vectors not corresponding to the masked time-frame. This results in a matrix of distractors $\mathbf{D} = (\mathbf{D}_1, ..., \mathbf{D}_{T_1})$ with $\forall i \in \{1, ..., T_1\} : \mathbf{D}_i \in \mathbb{R}^{M_D \times 2M_3}$. In practice, for both the wav2vec2 and XLS-R models, the number of distractors to sample is set to $M_D = 100$.

Using this, the loss can be computed as a combination between a contrastive loss, which evaluates the ability of the model to predict a contextual feature vector close to the corresponding quantized latent feature vector when compared to other quantized latent feature vectors from the same input, and a diversity loss, which aims at ensuring that the model makes use of as many speech codewords as possible. In mathematical formulation, denoting by $\mathcal{L}_c$ and $\mathcal{L}_d$ the contrastive and diversity losses respectively, the loss function of the model $\mathcal{L}$ can be written as follows, with $\alpha \in \mathbb{R}^+$ being a tuned hyperparameter that is fixed during training and set to $\alpha = 0.1$ in practice:

$$\mathcal{L} = \mathcal{L}_c + \alpha \mathcal{L}_d$$

In more detail, the contrastive loss $\mathcal{L}_c$ consists in a function that measures the ability of the model to identify, for each time-step, the correct quantized latent feature vector among a set of $M_D + 1$ candidates, composed of the M_D distractors and the true quantized representation. In mathematical formulation, this loss function can be written as follows, denoting by sim the cosine similarity operator (such that $\text{sim}(a, b) = a^\top b / \|a\|_2 \|b\|_2$)), with $\kappa \in \mathbb{R}^+$ a tuned hyperparameter, and indexing the matrices $\mathbf{D}_1, ..., \mathbf{D}_{T_1}$ by the rows:

$$\mathcal{L}_c = -\sum_{i=1}^{T_1} \left[\log \left(\exp \left(\frac{\text{sim}\left(C'_i, Q_i\right)}{\kappa} \right) \right) - \sum_{j=1}^{M_D} \log \left(\exp \left(\frac{\text{sim}\left(C'_i, D_{i,j}\right)}{\kappa} \right) \right) \right]$$

In short, one can see that at a given time-step, the loss decreases as the cosine similarity between the output of the context network and the corresponding, *ground-truth* quantized latent feature vector increases, and increases as the cosine similarity between the output of the context network and the distractors increases.

On the other hand, the diversity loss is designed to increase the use of a large number of speech codewords (hence of quantized representations). Denoting by B the total number of codebooks (set to $B = 2$ for the wav2vec2 and XLS-R models), by S the number of speech codewords in each codebook (set to $S = 128$ for the BASE version and to $S = 384$ for the LARGE version of the wav2vec2 and XLS-R models respectively), and by $\bar{p}_{b,s}$ the averaged softmax distribution for the speech codeword $s \in \{1, ..., S\}$ in codebook $b \in \{1, ..., B\}$, the diversity loss can be written as the sum of the entropies of these averaged softmax distribution among all speech codewords in each codebook, that is:

$$\mathcal{L}_d = \frac{1}{BS} \sum_{b=1}^{B} \sum_{s=1}^{S} \bar{p}_{b,s} \log \left(\bar{p}_{b,s} \right)$$

Once computed, this loss can be used as the criterion for an optimizing algorithm, which is taken to be the Adam algorithm introduced by Kingma et al. (2014) in both the wav2vec2 and XLS-R models. Using backpropagation, the model can be trained in a standard fashion. Metrics for evaluating this phase, on top of the losses, are essentially the accuracy of the context network to predict the correct quantized latent feature vector, as well as the number of codewords used in all codebooks.

As a short summary, the pre-training phase allows to learn meaningful contextual representations of the input raw speech data that can later be used for downstream tasks such as ASR. This process is based mostly on two encoders: a convolutional and a Transformer-based one, which help in learning both local and long-term dependencies in the data, as well as a masking scheme in the manner of the BeRT algorithm (Devlin et al. 2018). However, this process is highly dependant on a number of hyperparameters which can difficulty be set other than with heuristics: namely the learning rate, the parameters of the optimizing algorithm, the parameters of the quantization procedure (number of codebooks, number codewords, dimension of the codewords, temperature of the Gumbel-softmax operator, etc.), the parameters of the masking scheme (proportion of masked speech time-frames, number of consecutive speech time-frames to mask, etc.), as well as the hyper-parameters of the model architecture (kernel size and padding of the convolutional layers, number and input dimension of the Transformer encoder layers, etc.) and of the loss. A detailed account and discussion on the choice of hyper-parameters for different pre-training procedures is presented in Section 4.

Fine-tuning

The fine-tuning phase is the second training phase of both the wav2vec2 and XLS-R models, and consists in using only labelled speech data (*i.e.* with transcriptions) to solve the task of ASR in a more traditional way (*i.e.* omitting adjustments made to the architecture modules, in the form of standard end-to-end systems like DeepSpeech 2 (Amodei et al. 2016)). Ideally, this second phase requires little labelled data to attain good performance and fully leverages the representations learned during the first pre-training phase.

We display the architecture of the wav2vec2 fine-tuning phase in Figure 7 below. In the following, we also re-use all notations introduced above for the presentation of the pre-training phase, and omit the details which do not differ from it. Notice, for reference, the differences between Figure 7 and Figure 6.

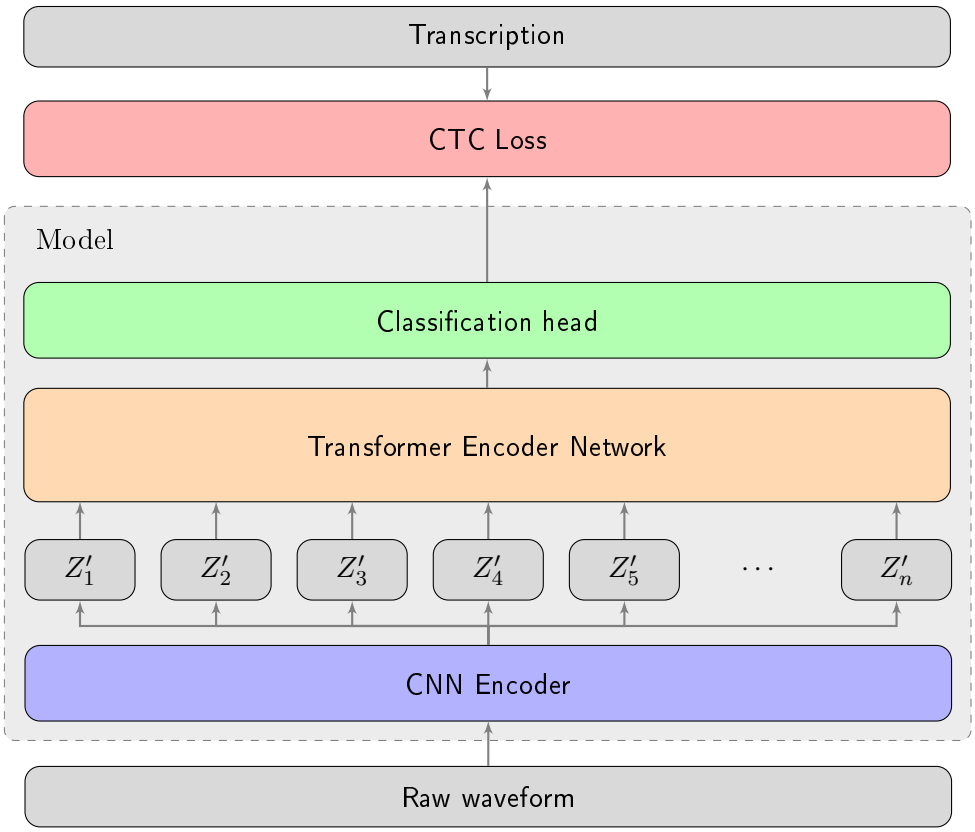


Figure 7: wav2ec2 fine-tuning model architecture

In short, the fine-tuning phase can be seen as a replication of the pre-training phase, without the masking scheme and quantization procedure, and using the Connectionnist Temporal Classification loss function (Graves, Fernàndez, et al. 2006) along with the transcriptions of input the speech audio waveform. To do this, the model takes again at input an digitally-encoded audio recording $\mathbf{X} = (x_1, ..., x_{T_0})$ of size T_0 with $\forall i \in \{1, ..., T_0\} : x_i \in \mathbb{R}$. Then, this audio waveform is fed to the same convolutional neural network encoder, which converts it to a sequence of latent feature vectors $\mathbf{Z} = (Z_1, ..., Z_{T_1})$ with $\forall i \in \{1, ..., T_1\} : Z_i \in \mathbb{R}^{M_1}$, which are then mapped to latent feature vectors that match the input dimension of the context network $\mathbf{Z'} = (Z'_1, ..., Z'_{T_1})$ with $\forall i \in \{1, ..., T_1\} : Z'_i \in \mathbb{R}^{M_2}$. As highlighted in Figure 7, these latent feature vectors are not quantized (since a contrastive loss is not longer used) nor masked. These are then fed to the Transformer encoder (*i.e.* the context network) to obtain meaningful contextual representations taking long-range dependencies into account. In mathematical formulation, in the same manner as in the pre-training phase, the context network yields a matrix of context vectors $\mathbf{C} = (C_1, ..., C_{T_1})$ with $\forall i \in \{1, ..., T_1\} : C_i \in \mathbb{R}^{M_2}$, which are again mapped to to a matrix $\mathbf{C'} \in \mathbb{R}^{T_1 \times 2M_3}$ using a simple linear layer.

Then, a classification head is added to the model with output size M_G corresponding to the size of the dictionary of graphemes (*e.g.* letters) used in the transcriptions, such that, for each time-frame, the context vectors are mapped to an output vector which encodes the logits relative to the probability of the considered time-frame to correspond to each unit of the dictionary. In

mathematical notations, this means that the classification head maps the context vector for each time frame classification head $C'_i \in 2M_3$ to an output vector $G_i \in M_G$. For instance, suppose that transcription is done in English, with a dictionary of 31 graphemes corresponding to the 25 letters of the alphabet along with a blank symbol and some punctuation elements. In this case, $M_G = 31$, and with the LARGE version of the wav2vec2 model, the classification head maps the context vector for each time frame $C'_i \in \mathbb{R}^{768}$ to an output vector $G_i \in \mathbb{R}^{31}$. In this setting, using a softmax operation at the output of the neural network, one can regard the vector $G_i \in \mathbb{R}^{31}$ as the relative probabilities of the considered speech time-frame to correspond to each grapheme in the dictionary.

The output of the fine-tuning model (both for wav2vec2 and XLS-R) thus consists in a matrix of vectors $\mathbf{G} = (G_1, ..., G_{T_1})$ such that $\forall i \in \{1, ..., T_1\} : G_i \in \mathbb{R}^{M_G}$, where each vector can be seen as the relative probabilities of a speech time-frame to correspond to each of the M_G units in the dictionary of graphemes. On top of that, the transcription of the speech audio recording is available and, following the pre-processing guidelines, contains only graphemes included in the dictionary. This data is then sufficient to compute the loss function and update the weights of the model depending on this loss and an optimizing algorithm.

To do this, the wav2vec2 and XLS-R models use the Connectionnist Temporal Classification (CTC) loss function (Graves, Fernàndez, et al. 2006), which allows to evaluate how well the sequence of vectors $(G_1, ..., G_{T_1})$ and the transcription align despite the sequential nature of the output and without ever requiring manual annotation of the graphemes pronounced at each speech time-frame. In short, CTC bridges the gap between the sequential nature of the transcription and the sequential nature of speech. This loss function, originally introduced for the task of recognition of written text on images, helps in shortcoming two major issues in ASR: first, it is highly time-consuming to manually annotate the pronouciation of individual graphemes in speech recordings; and then multiple time-frames may correspond to the same grapheme depending on the variations in pronouciation and elocution of the person speaking when recording the audio.

The CTC loss function is based on two components: a text encoder and and alignment path finder. On the one hand, the text encoder manages issues related to the repetition of graphemes in the sequence of embeddings at the output of the neural network. To do this, a pseudo-character is introduced in the dictionary before training, which is denoted as the blank character (different from the white space) and which marks the switch between two graphemes in a sequence of embeddings. This character is usually denoted by `_` in practice. Then, this pseudo-character can be predicted as one of the elements in the dictionary, and the CTC loss function encodes the text using two rules: first, the deletion of repetitions not separated by a blank character, and then, the deletion of blank characters. Suppose, for instance, that we wish to encode the sequence of characters: `CC_AAAT`. Then, during the first step, the repetitions not separated by a blank character are deleted, which yields the sequence `C_AT`. In the second step, all blank characters are deleted, giving the word `CAT`. In the same vein, suppose this time that we wish to encode the sequence of characters: `AAP_PP_LLLE_`. Then, during the first step, the repetitions not separated by a blank character are deleted, which yields the sequence `AP_P_LE_`. Finally, all blank characters are deleted, giving this time the word `APPLE`. Notice, in particular, that this encoding process is not injective, or, in other words, that multiple sequences can yield the same final word.

Then, the alignment path finder uses this text encoder to compute the loss. In short, the combination of the sequence of time-steps and number of units in the dictionary yield a large amount of potential alignment paths, where an alignment path simply consists of a sequence of graphemes. In the CTC loss calculation, these alignment paths are given scores depending on the logits at the output of the neural network, such that the score for an alignment path is given by the multiplication of the posterior associated to the desired grapheme in the dictionary at each time-step. In Figure 8 below, an example of the computation of the CTC loss is given, where the number of time-steps at the output of the neural network is $T_1 = 6$, and the dictionary is set to a simple dictionary with only 4 graphemes: $\{\texttt{A}, \texttt{B}, \texttt{C}, \texttt{\_}\}$, so that $M_G = 4$. In this Figure, the logits corresponding to each grapheme are written at the bottom of the cells. Notice that, since these logits are the result of a softmax operation, the column-wise sum of all logits is always equal to one. Finally, the alignment path with maximum score is drawn with lines.

In this Figure, the score for an alignment path can be computed as the multiplication of the probabilities associated to each grapheme in the alignment path at each time-step. For instance, the score of the path `C__AB_` is equal to $0.61 \times 0.78 \times 0.62 \times 0.56 \times 0.59 \times 0.91 \approx 0.089$. The score of the path `AA_CC_` is equal to $0.24 \times 0.07 \times 0.62 \times 0.06 \times 0.22 \times 0.91 \approx 1.25 \times 10^{-4}$. Following this procedure, the score of all alignment paths can be computed.

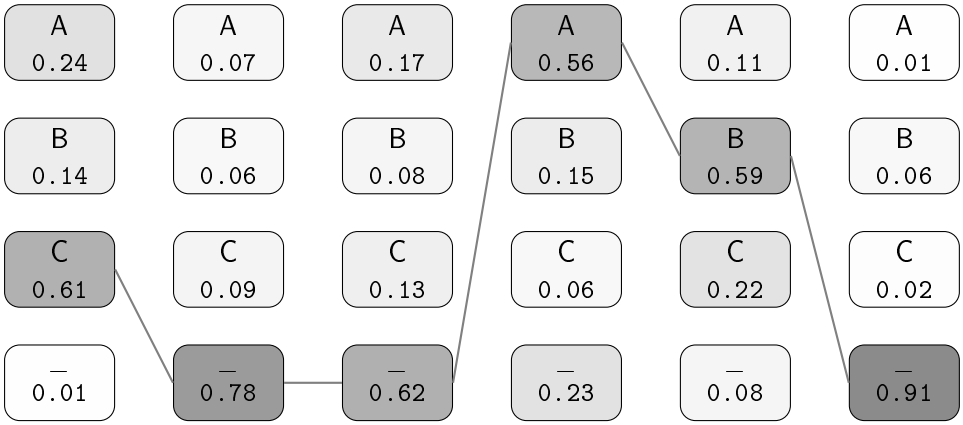


Figure 8: Example of computation of the CTC loss function

Then, the CTC loss function is computed as the negative sum of the log-scores over all alignment paths which, after encoding, yield the groud-truth transcription. For instance, in the example in Figure 8, suppose that the ground-truth transcription is `CAB`. Then, as the encryption operator $\mathcal{B}$ is not injective, multiple alignment paths are recuded to the same ground-truth transcription after encoding. For instance, the alignment paths $\{\texttt{C}, \texttt{\_}, \texttt{A}, \texttt{\_}, \texttt{\_}, \texttt{B}\}$, $\{\texttt{C}, \texttt{A}, \texttt{\_}, \texttt{\_}, \texttt{B}, \texttt{B}\}$ and $\{\texttt{\_}, \texttt{C}, \texttt{A}, \texttt{\_}, \texttt{\_}, \texttt{B}\}$ all yield `CAB` as final transcription after encoding. Hence, in this example, the loss is computed as the negative sum of the logs of the scores over all alignment paths whose final transcription after encoding is `CAB`. In mathematical notation, denoting by g_t the ground-truth transcription ((*e.g.* $g_t = \{\texttt{C}, \texttt{A}, \texttt{B}\}$ in the example above), $\mathcal{B}^{-1}(g_t)$ the set of all alignment paths whose encoding yields g_t, s the function which maps a letter in an alignment path to its logit, and indexing the alignment paths character-wise, the CTC loss function $\mathcal{L}_{CTC}$ can be expressed as:

$$\mathcal{L}_{CTC} = -\sum_{p \in \mathcal{B}^{-1}(g_t)} \log\left(\prod_{i=1}^{T_1} s(p_i)\right) = -\sum_{p \in \mathcal{B}^{-1}(g_t)} \sum_{i=1}^{T_1} \log\left(s(p_i)\right)$$

Once computed, this loss can then be used as the criterion for an optimizing algorithm, which is taken to be the Adam algorithm again (Kingma et al. 2014) in both wav2vec2 and XLS-R. Using backpropagation, the model can be trained in a standard fashion. Metrics for evaluating this phase are essentially composed of the CTC loss function and the utterance error-rate (UER), often taken to be a word error-rate (WER) or phoneme error-rate (PER) (defined below).

Briefly, the fine-tuning phase allows to perform the task of ASR in a supervised manner leveraging the meaningful contextual representations of the input raw speech data learnt in the pre-training phase. As such, it solves the task of ASR in a standard way, using the CTC loss function and removing the masking scheme and quantization procedure from the network architecture. However, just as much as for the pre-training phase, this process is highly dependant on a number of hyperparameters which can difficulty be set other than with heuristics. Among others, the learning rate and its scheduler, the parameters of the optimizing algorithm, the layerdrop and batch-size have a decisive impact on the convergence of the model. A detailed account and discussion on the choice of hyper-parameters for different fine-tuning experiments is presented in Section 4.

Decoding

The decoding phase can be seen as the last phase in using the wav2vec2 and XLS-R models. In short, decoding is used for validation during the fine-tuning phase, as well as after the training of the model is finished for evaluation and use of the ASR system on new data. As displayed in the last section, through the computation of the CTC loss function, the fine-tuning phase already makes the link between the logits outputted by the model at each time-frame (also known as the posteriogram), and the transcription of the speech audio. However, in this phase, the link depehnds heavily on the knowledge of the ground-truth transcription, which helps in finding the alignment paths that correspond to that transcription among all possible alignment paths, which themselves are needed in the computation of the loss.

The task of decoding is based on the same ideas developed during the computation of the CTC loss, but is done with no information about the ground-truth transcription. In short, it is a task of evaluation, whose goal is both to measure the ability of the model to find the ground-truth transcription using only the audio data, and to use the model for the task of ASR on data for which the user does not have access to ground-truth transcriptions. Hence, the task of decoding can be summarised as the task of finding the most likely transcription among all possible transcriptions.

Recall that, after a pass through all the layers in the neural network, the data which can be used to determine the transcription consists in a matrix $\mathbf{G}$, also known as the posteriogram, such that $\mathbf{G} = (G_1, ..., G_{T_1})$ where each output vector $G_i \in \mathbb{R}^{M_G}$ corresponds to the logits over all M_G graphemes in the dictionary. For instance, with a dictionary of 31 graphemes corresponding $M_G = 31$, and an audio speech file of 15.31 sampled at 16kHz, the input vector would be a vector of size $15.31 * 16000 = 244,960$, resulting, following the same calculation done in the previous

sub-section, in 765 speech time-frames, thus in a posteriogram $\mathbf{G} \in \mathbb{R}^{765\times 31}$ with a vector of dimension 31 for each individual time-frame.

Finding the transcription with maximum output probability among all possible transcriptions in the posteriogram is a combinatorial problem that cannot be solved in reasonable amount of times with long sequences (*i.e.* audio speech recordings lasting multiple seconds). We display why this is the case in Figure 9. In this Figure, the size of the dictionary is set to $M_G = 3$ (where the dictionary is $\{\texttt{A}, \texttt{B}, \_\}$), with only $T_1 = 2$ speech time-frames. Hence, the model admits only five possible transcriptions: `A`, `B`, `AB`, `BA` and a transcription corresponding to silence. The sequences corresponding to the transcription `A` are $\{\{A, \_\}, \{\_, A\}, \{A, A\}\}$, the sequences corresponding to the transcription `B` are $\{\{B, \_\}, \{\_, B\}, \{B, B\}\}$, the sequence corresponding to the transcription `AB` is $\{\{A, B\}\}$, the sequence corresponding to the transcription `BA` is $\{\{B, A\}\}$ and finally the sequence corresponding to the silence transcription is $\{\{\_, \_\}\}$. Using the logits displayed in the figure, it follows that the probability of the transcription to be `A` is $0.32\times 0.47+0.32\times 0.49+0.51\times 0.47 \approx 0.55$; the probability of the transcription to be `B` is $0.17\times 0.04+0.17\times 0.49+0.51\times 0.04 \approx 0.11$; the probability of the transcription to be `AB` is $0.32 \times 0.04 \approx 0.01$; the probability of the transcription to be `BA` is $0.17 \times 0.47 \approx 0.08$; and the probability of the silence transcription is $0.51 \times 0.49 \approx 0.25$. Hence, the transcription with maximum probability is `A` with 55%.

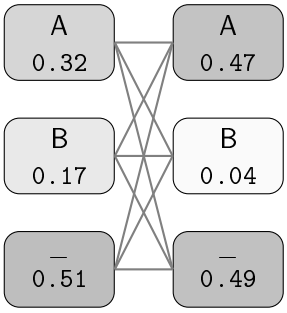


Figure 9: Example of posteriogram in the decoding phase

This example sheds light on two key difficulties when performing decoding. First, because the encoding operator $\mathcal{B}$ is not injective, the transcription with maximum probability is not necessarily the one made of the grapheme with maximum probability at each time-frame; as a matter of fact, in our example, the most likely transcription is `A` although the grapheme `A` is never the grapheme with maximum probability, across all time-frames, Then, following this observation, it is straigtforward to see that finding the transcription with maximum probability requires the computation of the probability associated to all alignment path; hence is has a complexity in $\mathcal{O}(M_G^{T_1})$, which makes it intractable in practice with a large dictionary (*e.g.* $M_G = 31$ for English) and with long audio recordings. To overcome these two problems, multiple approaches have been put forward. We give a brief overview of these approaches in this section, without delving too much in the mathematical theory that allows them to work in practice.

Greedy decoding The first type of decoding is the greedy decoding, introduced by Graves, Fernàndez, et al. 2006 in their original paper on the Connectionnist Temporal Classification loss, and which amounts to taking the grapheme with maximum probability at each speech time-frame. As we have seen in the example of Figure 9, this method is very prone to errors, as it fails

to take into account the fact that multiple alignment paths can result in the same transcription (in the example, the greedy decoding algorithm would output the silence transcription, although the transcription `A` has a score twice as high). However, the operation only requires only performing an argmax operation at each time-step on the output of the network, and has therefore a complexity in $\mathcal{O}(T_1)$, rendering it very efficient.

Beam-search decoding In order to improve on the heuristic made by the greedy decoding algorithm, the beam-search decoding algorithm has been heavily used both for evaluating and using the models in practice, not only in ASR but also for NLP and OCR tasks. In short, the beam-search decoding algorithm works by selecting multiple alternatives at each speech time-frame, where the number of alternatives retained is called the beam width, and sequentially dropping the alternatives with least conditional probability to end up corresponding to a transcription with high score. In other words, this implies that, with a beam width of 3, the beam search decoder will first consider all the graphemes for the first speech time-frame, and then retain the 3 best sequences among all possible sequences of two graphemes. Then, on the third speech time-frame, the decoder will compute the score associated to all sequences made of the three two-grapheme sequences kept in the second step, and retain only the three three-grapheme long sequences that have the highest score. The same operation is then repeated until the final speech time-frame, at which time the transcription is chosen depending on the transcription with highest probability looking only at the three final sequences. Hence, beam-search decoding sacrifices completeness by not exploring all possible sequences, but improves largely over complete decoding in terms of complexity. Denoting by M_W the beam width, the beam-search decoding algorithm has a complexity in $\mathcal{O}(M_W M_G^2 T_1)$. In practice, the beam width is taken to have a relatively high value (*e.g.* $M_W = 200$).

Beam-search decoding with language model The beam-search decoding algorithm can also be used in conjuction with a language model. Introduced by A. S. Hannun et al. (2014) for the task of ASR, this decoding strategy relies on the same principle as the beam-search decoding, but chooses the sequences to be studied recursively not only using the score outputted by the neural network but also by penalizing the scores depending on the probability of appearance of the words in the transcriptions resulting from the sequences. In other words, beam-search decoding with a language model amounts to applying a prior on the probability of the sequences, where the prior depends exclusively on a language model and hence on the properties of written kanguage rather than spoken language. As such, beam-search decoding with a language model helps in making the sequences kept in the beam-search algorithm more likely with regards to written text, hence often to improve the quality of the transcription. However, this prior knowledge may also be detrimental to the model's performance, as it might make the beam-search decoder avoid sequences with rare words or words that have not been used to train the language model, despite them having a larger score. In practice, the language models used in beam-search decoding with language models are n-gram models or Transformer models, which we do not detail here. A short overview of the types of language models is however available in the appendix.

Fully differentiable decoders Finally, although recent architectures have been named end-to-end as opposed to the older systems that worked by combining multiple bricks (conversion of the raw audio to spectograms, pre-processing, etc.), the decoding phase remains external to the

model in the sense that it does not leverage any learning algorithm and is rule-based, both for the greedy and beam-search decoders. In this vein, some approaches, notably that of Collobert et al. (2019) have tried to implement a decoding algorithm that retains differentiability and hence can be integrated in the training of the whole neural network. However, these approaches have so far scarcely been used and are not implemented neither in wav2vec2 nor XLS-R.

In short, the decoding phase consists in making the link between the output produced by the neural network (a posteriogram with the logits corresponding to each grapheme at each speech time-frame) and a readable transcription. As highlighted in the example of Figure 9, this task is particularly arduous due to the computational complexity that a complete decoding requires. As a response to this, multiple approaches have been explored, notably the easy-to-implement greedy decoder, and the beam-search decoder (with or without language model), which both eable quick computation of a transcription (hence short evaluation times), at the cost of potentially missing good transcriptions. In the rest of this report, we use the greedy and 4-gram beam-search decoders in our experiments.

Evaluation

Finally, we describe the evaluation metrics that we use for our experiments on the wav2vec2 and XLS-R models. Traditionally, ASR models are evaluated using the word-error rate (WER) metric, which corresponds to the number of substitutions, deletions and insertions in the transcription given by the model when compared to the ground-truth transcription, divided by the total number of words in the original (ground-truth) transcription. As such, the domain of definition of the word error-rate is $\mathbb{R}^+$, as the model may, for instance, output a transcription with more additions than there are words in the ground-truth transcriptions. As such, in a rigorous view, it cannot be read as a percentage and should not be interpreted as such. However, in practice, the word error-rate rarely gets over the unit value, and is thus often seen as a percentage, denoting the number of *wrong* words in the transcription over the total number of words in the ground-truth transcription. In mathematical formulation, the word error-rate is defined as:

$$\text{WER} = \frac{\text{substitutions} + \text{deletions} + \text{insertions}}{\text{total number of words}}$$

For instance, suppose that the model is given for input an audio recording of a person pronouncing the sentence `THE CAT IS IN THE GARDEN AND LOOKS AT THE WINDOW`. Suppose also that, after passing through the neural network and the decoder, the transcription given by the network is `THE CAT EASING THE GARDEN END LOOKS AT THE WIND DOE`. Then, one can see that there is are three substitutions (`EASING` for `IS`, `END` for `AND`, and `WIND` for `WINDOW`), one deletion (`IN`) and one addition (`DOE`). The total number of words in the ground-truth sentence is 11. Hence, the WER for this transcription is $(3 + 1 + 1)/11 \approx 0.45$.

Other similar evaluation metrics are often used in the litterature. These usually take the form of an utterance error rate, where an utterance to a spoken unit, *e.g.* a word, a character, a phoneme, a word-piece, etc. With this rather general notion at hand, one can define the utterance error-rate (UER) int the same manner as with the word error rate, as:

$$\text{UER} = \frac{\text{substitutions} + \text{deletions} + \text{insertions}}{\text{total number of utterances}}$$

In the rest of this report, results will be given essentially in terms of word error-rate (WER) and character error-rate (CER), which are the two metrics most often used by the literature on ASR systems. Additionally, as highlighted in Section 2, some references to phoneme error-rates are made as some datasets are particularly prone to using phonemes rather than words or characters, and as some research work has recently focused on phonemes rather than characters. It is, however, important, to note that while word error-rates can be compared across models on the same evaluation dataset, metrics using different types of utterances cannot be compared even if the evaluation dataset is the same, as the length of the utterances can have a strong impact on the error-rate.

3.3 Technical challenges

As highlighted in the previous subsections as well in the experimental parts of the report, pre-training and fine-tuning a wav2vec2 or XLS-R models are arduous tasks as the performance of the model is highly dependent on the choice of hyper-parameters. As a matter of fact, as is shown in Section 4, a small change in a single hyperparameter like the batch-size can have a strong impact on the convergence of the model.

These difficulties, which, although a vast part of the research on hyperparameter search is heuristic, are of the scientific kind, as they are deeply linked with the convergence properties of the training algorithm used for estimating the optimal weights of the neural network, and with the mathematical properties of deep neural networks.

In this brief section, we highlight that training these models is all the more difficult that technical challenges are added on top of the scientific questions. As a matter of fact, we observe in our experiments that pre-training a BASE version of a wav2vec2 or XLS-R model on 1000 hours of speech data requires around two days of computation on 64 V100 GPUs (*i.e.* 3,072 hours of computation on a single V100 GPU), and pre-training the LARGE version of these models on the same corpus requires around three days of computation on 128 V100 GPUs (*i.e.* 9,216 hours of computation on a single V100 GPU).

Comparably, fine-tuning the BASE version of the wav2vec2 or XLS-R requires between one and two days of computation on 8 V100 GPUs (*i.e.* between 192 and 384 hours on a single V100 GPU), and fine-tuning the LARGE version between one and two days of computation on 16 V100 GPUs (*i.e.* between 384 and 768 hours on a single V100 GPU). In practice, due to scheduling operations on the supercomputers used during the internship, the actual time to train these models took about twice the time in GPU usage (as instances were not always available). For instance, pre-training a LARGE version of the wav2vec2 model took, on average, about a week of time due to the limitations of the training times on the supercomputer cluster.

More importantly, although the resources required to train these models were made available from the start of this internship, the limited availability of such an important number of instances, as well as the duration of the training largely hinder the possibility of searching for optimal hyperparameters via operations like a grid-search or even more efficient procedures for hyperparameter search. On top of this, the training of such models requires great care, as resources are both costly, in limited availability, and as training models using such important amounts of computations has an environmental cost.

4 Using end-to-end backbone ASR models in practice

In this section, we focus on the results published by the authors of the wav2vec2 and XLS-R models, and perform experiments to guide the use of these models in practice. In short, we start by the more arduous task of pre-training models using the same set of hyparameters of the authors of the papers, and try to reproduce their results in terms of accuracy. Following this, in order to evaluate the quality of these pre-trained models, we reproduce their fine-tuning training phases, using the pre-trained models as basis, and compare the results obtained to that of the authors. Finally, we focus on the language modelling and decoding phase, and illustrate the dependence between the quality of the transcription and the generalization capabilities of the language model used in decoding via experiments on the pre-trained and fine-tuned models.

4.1 Pre-training mono-lingual ASR models

In our first mono-lingual approach, we try to reproduce the results detailed by Baevski et al. (2020) for the wav2vec2 model on the LibriSpeech dataset. To do this, we pre-train the wav2vec2 model from scratch using the same set of hyperparameters. In detail, we pre-train both the BASE and LARGE versions of the model on the original LibriSpeech training corpus consisting in roughly 960 hours of speech from audiobooks recordings, using the 5 hours of the development other validation subset for validation. In all experiments carried out below, we use the architecture detailed in Section 3 and state the hyperparameters that are changed.

For the BASE version of the model, Baevski et al. (2020) perform the pre-training phase by using batches composed of 15.6 seconds (corresponding to 250,000 samples) cropped from the raw audio data recordings, batched together so as to not exceed 1.4 million samples per GPU. In total, on all 64 V100 GPUs, the batch-size is thus approaching $(1,400,000/250,000) \times 15,6 \times 64 \approx 1.55$ hours. The learning rate is set to follow a linear decay with a warmup consisting in the first 8% of updates, reaching a peak of 5×10^{-4}. The model is trained with a weight of $\alpha = 0.1$ for the diversity loss, with a Gumbel softmax temperature parameter annealed from 2 to a minimum of 0.5 by a factor of $1 - 5 \times 10^{-6}$ at each update, and the temperature in the contrastive loss is set to $\kappa = 0.1$. On top of this, a L2 penalty term is added to the activations of the final layer of the feature encoder to ensure regularization, the gradients a scaled down by a factor of 10, and layers are randomly dropped at each update at a rate of 5%. The number of distractors is set to 100, the number of codebooks to 2 and the number of codewords in each codebook to 320, where each of these codewords has dimension 128. The model is trained on 64 V100 GPUs for 400,000 updates (where training time varies depending on the availability of NVLinks).

For the LARGE version of the model, the authors perform the pre-training phase by using this time batches composed of 20.0 seconds (corresponding to 320,000 samples) cropped from the raw audio data recordings, batched together so as to not exceed 1.2 million samples per GPU. In total, on all 128 V100 GPUs, the batch-size is thus approaching $(1,200,000/320,000) \times 20.0 \times 128 \approx 2.67$ hours. The learning rate is set to follow a linear decay with a warmup consisting in the first 8% of updates, reaching a peak of 5×10^{-5}. The model is trained with a weight of $\alpha = 0.1$ for the diversity loss, with a Gumbel softmax temperature parameter annealed from 2 to a minimum of 0.5 by a factor of $1 - 5 \times 10^{-6}$ at each update, and the temperature in the contrastive loss is set to $\kappa = 0.1$. On top of this, a dropout of 0.1 is used in the Transformer part of the model at the

output of the CNN feature encoder, and layers are randomly dropped at each update at a rate of 20%. The number of distractors is set to 100, the number of codebooks to 2 and the number of codewords in each codebook to 320, where each of these codewords has dimension 384. The model is trained on 128 V100 GPUs for 250,000 updates (about three days).

Details regarding the accuracy and loss values obtained by the authors during the pre-training phase have, to the best of our knowledge, not been released publicly. Hence, we display in the following figures and tables the results obtained, keeping in mind that the discriminative performance of these features can only be truly measured after the fine-tuning phase. Nonetheless, we observe a smooth convergence in our models towards an accuracy of 65% to 75%, both on the training and validation datasets. Figure 10 displays the accuracy of the pre-training version of the BASE wav2vec2 model during training, where the accuracy is measured by the model's ability to output contextual encodings that are closer to the ground-truth speech codeword than others in the codebooks. As can be seen in the figure, the accuracy of the model reaches a peak of 67.6%, attained at the end of training.

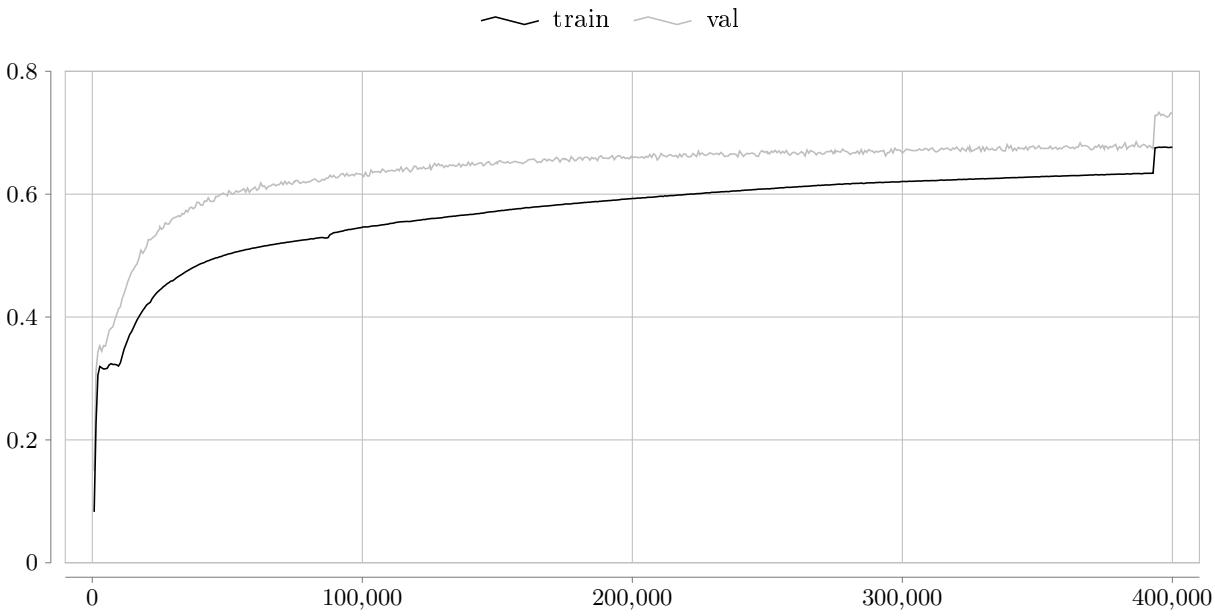


Figure 10: Accuracy of the BASE version of the wav2vec2 model during pre-training on the LibriSpeech dataset

As Figure 10 shows, small jumps (steep increases or decreases in the metric's value) can be observed roughly around steps 10,000 and 90,000. Heuristically, looking at the pre-training metrics, one can hypothesise that these small jumps are due to threshold effects on the diversity loss, hence on the number of speech codewords used in each codebook by the model, which is a hyperparameter directly linked to the Gumbel-softmax temperature parameter, which is annealed during training from 2 to 0.5. As a matter of fact, as Figure 11 displays (on the next page), the code perplexity, which measures the perplexity of the Gumbel-softmax distribution over all possible values, hence indicates the number of codewords used in practice by the model, observes jumps at the same steps of the pre-training phase.

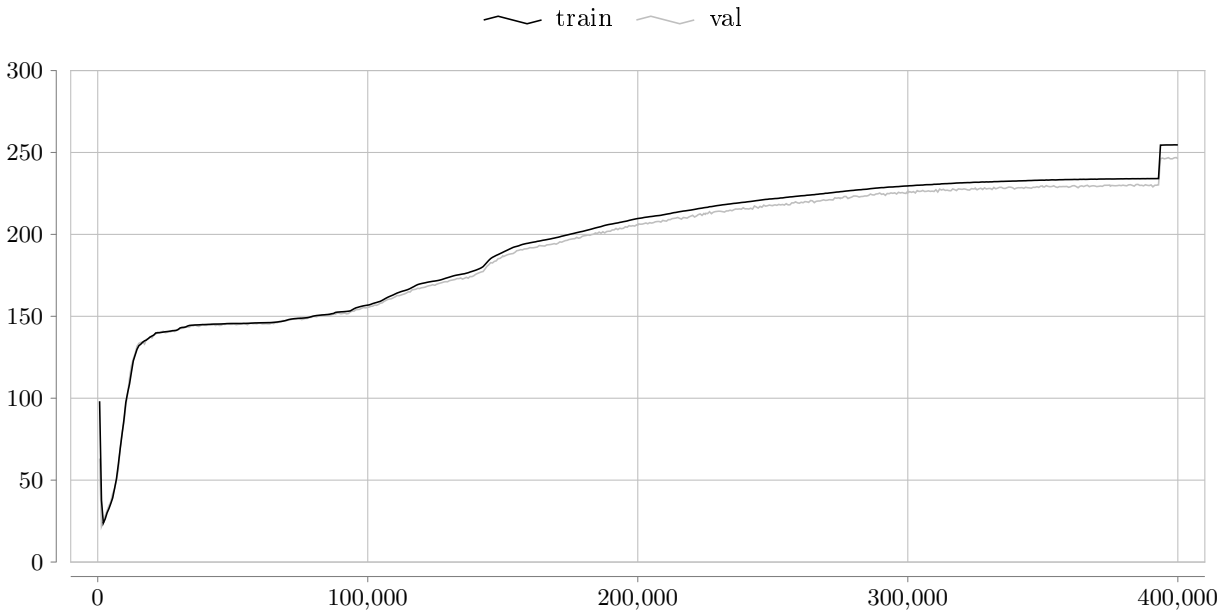


Figure 11: Code perplexity of the BASE version of the wav2vec2 model during pre-training on the LibriSpeech dataset

In a similar fashion, Figure 12 displays the accuracy of the pre-training version of the LARGE wav2vec2 model during training, which reaches a peak of 67.6% at the end of training phase. The same continuous increase in accuracy can be observed as in the BASE version of the model.

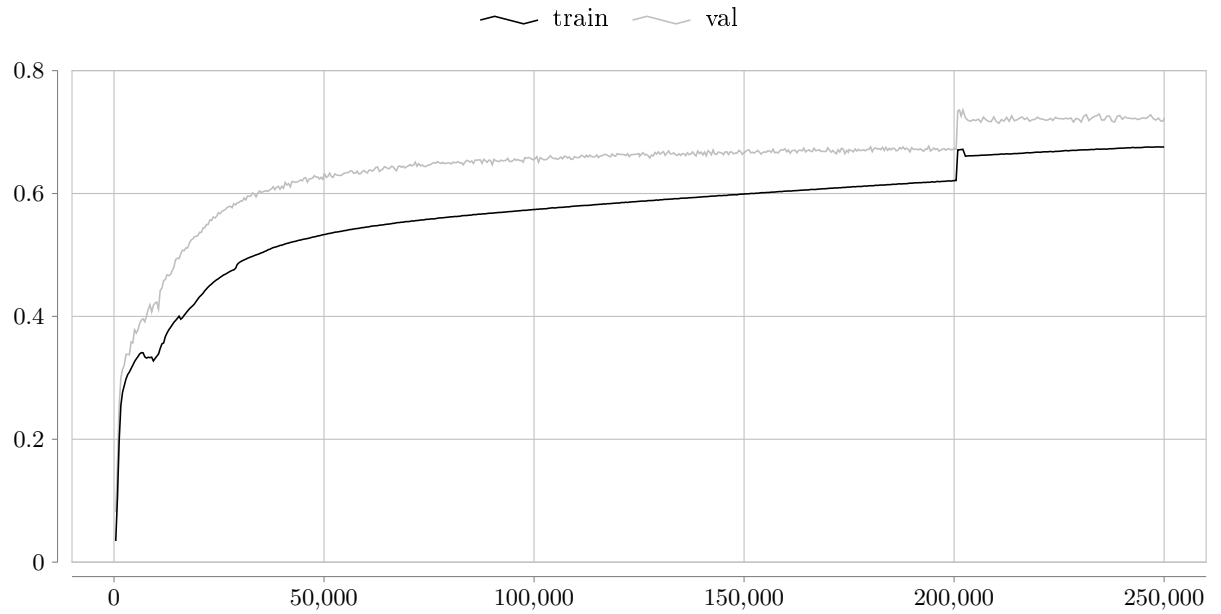


Figure 12: Accuracy of the LARGE version of the wav2vec2 model during pre-training on the LibriSpeech dataset

As Figure 12, small jumps can again be observed in the accuracy, this time at steps 8,000 and 30,000. Again, heuristically, looking at the pre-training metrics, one can hypothesise that the same mechanic is at work: *i.e.* that these small jumps are due to different threshold effects on the diversity loss, where these would be linked to the annealing scheme used for the Gumbel-softmax temperature parameter. As a matter of fact, as Figure 13 shows, the code perplexity observes again the same two jumps during pre-training, at roughly the same steps.

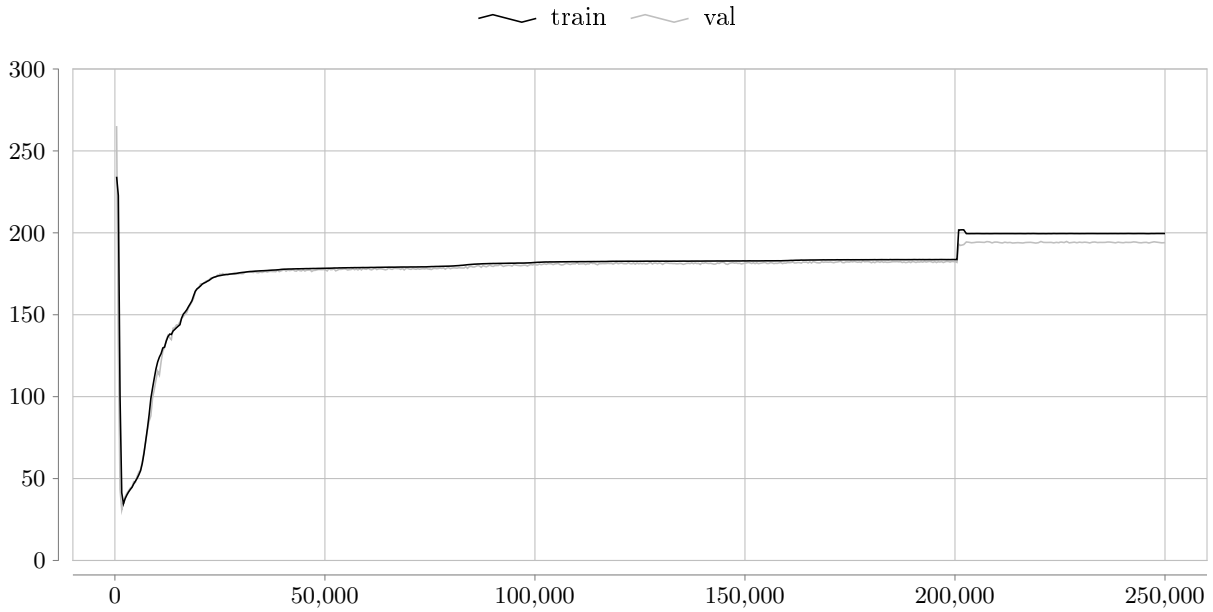


Figure 13: Code perplexity of the BASE version of the wav2vec2 model during pre-training on the LibriSpeech dataset

A summary of these results is displayed in Table 2, which indicates, both for the BASE and LARGE versions of the model, the final accuracy obtained at the end of the pre-training phase, as well as code perplexity. We see that the accuracy is slightly higher for the LARGE version of the model, but the significance of this difference remains open to discussion. The same can conclusion can be made for the code perplexity.

Version	Accuracy		Code perplexity	
	train	dev	train	dev
BASE	67.6%	73.1%	254.6	246.3
LARGE	67.6%	72.3%	199.6	194.1

Table 2: Final accuracy and code perplexity of the BASE and LARGE versions of the wav2vec2 model during the pre-training phase

Two heuristic observations can be inferred from this first pre-training phase about the hyper-parameters of the wav2vec2 model. First, in our experiments, pre-training accuracy was highly sensitive to the setting of the learning rate, with learning higher or lower by a factor of 10 (*e.g.* a learning rate of 5×10^{-3} or 5×10^{-5} for the BASE model, or 5×10^{-4} or 5×10^{-6} for

the LARGE model) resulting in accuracies more than 10% below the one obtained on the final model presented above in absolute terms. Then, the Gumbel-softmax temperature parameter also has a strong influence on the final quality of the model, where, as can be seen in the Figures above, the annealing rate of this hyperparameter plays a decisive role in the number of speech codewords used by the quantizer of the model, which in turns affects the expressive properties of the contextual representations learned by the model.

Finally, and perhaps most importantly, it remains particularly arduous to understand the impact of this phase on the quality of the transcription of the model once all training phases have been completed. As a matter of fact, apart from these indirect metrics measuring the accuracy of the model and its code perplexity, there is, to our knowledge, no way to assess the quality of the contextual representations with regards to the task of ASR other than by fine-tuning the model. However, as exposed in Section 3, performing both the pre-training and fine-tuning phases to evaluate a model is costly in terms of computation, thus hyper-parameter search is hardly achievable.

4.2 Fine-tuning mono-lingual ASR models

Following these conclusions, we go on by trying to reproduce the results detailed by Baevski et al. (2020) for the wav2vec2 model on the LibriSpeech dataset in the fine-tuning phase. To do this, we fine-tune the wav2vec2 model from scratch using the same set of hyperparameters as the authors. In greater detail, we perform fine-tuning for both the BASE and LARGE versions of the model on the original subsets of the LibriSpeech training corpus consisting in respectively 1, 10 and 100 hours of speech from audiobooks recordings, using the 5 hours of the development clean and other validation subsets for validation, and the 5 hours of the testing clean and other subsets for evaluation. In all experiments carried out below, we use the architecture detailed in Section 3 and state the hyperparameters that are changed.

For the BASE version of the model, Baevski et al. (2020) perform the fine-tuning phase by using batches built from the raw audio extracts and composed of a maximum total of 3,200,000 samples per GPU. In total, on all 8 V100 GPUs, the batch-size is thus approaching $(3,200,000 \times 8)/16,000 \approx 0.44$ hours. The learning rate follows a tri-state schedule where it is first warmed-up for the first 10% of updates, then held constant for 40% of updates, and finally linearly decayed for the last 50% of updates. Its peak value, as well as the layer drop probability, time-step masking probability, channel-masking probability and number of total updates depend on the size of the corpus considered, and are detailed in Table 3 below.

Corpus	Learning rate	Layer-drop	Time-step mask.	Channel mask.	Updates
1 hour	1×10^{-4}	0.05	0.75	0.256	13,000
10 hours	1×10^{-4}	0.05	0.65	0.256	20,000
100 hours	1×10^{-4}	0.05	0.50	0.512	50,000

Table 3: Fine-tuning hyperparameters of the BASE version of the wav2vec2 model

For the LARGE version of the model, the authors perform the fine-tuning phase by using batches built from the raw audio extracts and composed of a maximum total of 1,280,000 samples per GPU. In total, on all 24 V100 GPUs, the batch-size is thus approaching $(1,280,000 \times$

$24)/16,000 \approx 0.53$ hours. The learning rate follows a tri-state schedule where it is first warmed-up for the first 10% of updates, then held constant for 40% of updates, and finally linearly decayed for the last 50% of updates. Its peak value, as well as the layer drop probability, time-step masking probability, channel-masking probability and number of total updates depend on the size of the corpus considered, and are detailed in Table 4 below.

Corpus	Learning rate	Layer-drop	Time-step mask.	Channel mask.	Updates
1 hour	3×10^{-5}	0.1	0.75	0.256	13,000
10 hours	3×10^{-5}	0.1	0.65	0.256	20,000
100 hours	1×10^{-4}	0.1	0.50	0.512	50,000

Table 4: Fine-tuning hyperparameters of the LARGE version of the wav2vec2 model

In the following, we display figures of the CTC loss functions obtained after training the models a the 10 hours corpus of the LibriSpeech dataset, using the training scheme described in Section 3. Figure 14 displays the loss of the fine-tuning version of the BASE wav2vec2 model during training on the 10h subset of the LibriSpeech training data (which is the main corpus considered as it is of highest interest for this internship with regards to the amount of data available for internal corpora), where the loss is the CTC loss described in section 3.

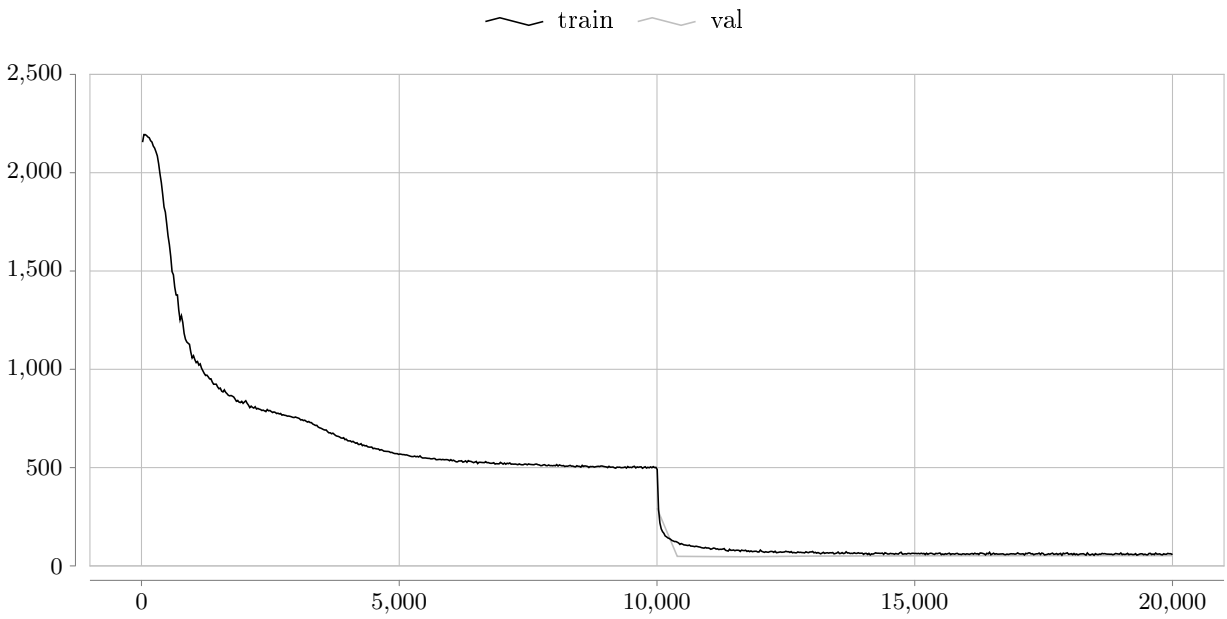


Figure 14: CTC loss of the BASE version of the wav2vec2 model during fine-training on the LibriSpeech 10h sub-dataset

Likewise, the loss of the of the LARGE version of the wav2vec2 model during fine-tuning is displayed in Figure 15. These two losses show two phases: a first phase a sharp decrease that ends with a stabilisation of the loss, followed by another sharp decreases. The first phase corresponds to the warmup phase. which lasts for the first 10,000 iterations, and which corresponds to a phase during which only the CNN Encoder is updated, such that the Transformer context network is

only updated in the second phase of the fine-tuning period. Values for the CTC loss during the fine-tuning phase have, to the best of our knowledge, not been released publicly by the authors. For comparison, we thus mainly focus on the word error-rate and character error-rate values, which are not reported here but left for the next section on language modelling and decoding.

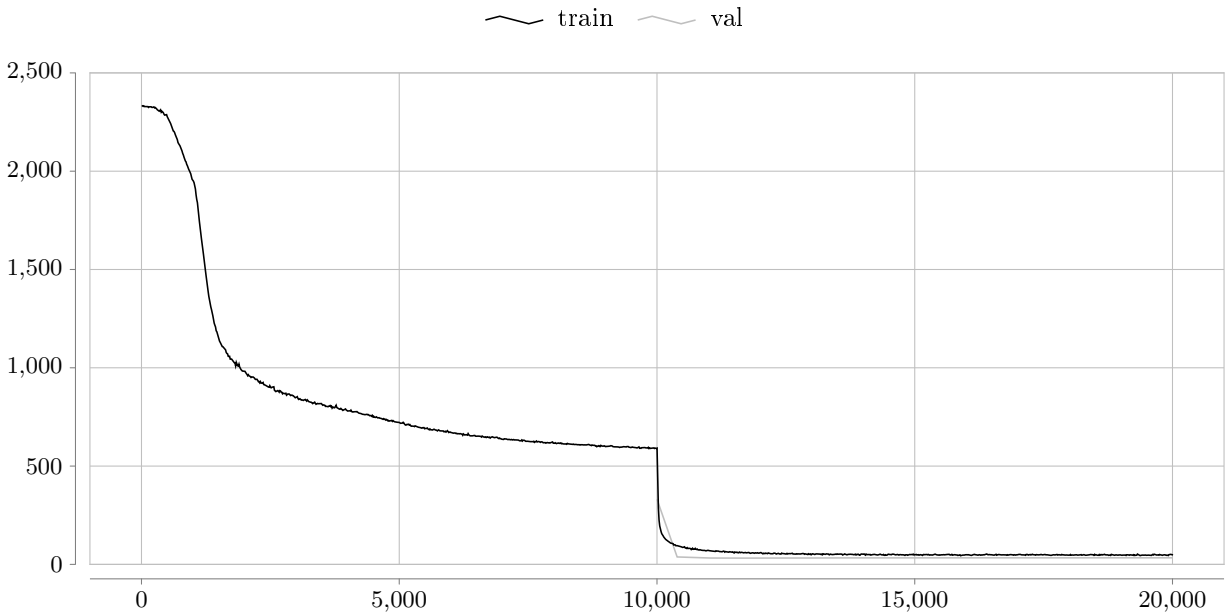


Figure 15: CTC loss of the LARGE version of the wav2vec2 model during fine-training on the LibriSpeech 10h sub-dataset

The convergence of the 1 hour and 100 hours subsets of the LibriSpeech corpus follow, roughly, the same pattern as the one observed above for the 10 hours corpus. For readability of this report and because these do not add much content to the conclusions we draw, the figures displaying these losses are placed in the appendix, along with figures of the learning rate schedules.

4.3 Language modelling and decoding

As our final step towards reproducing the results obtained by Baevski et al. (2020), we evaluate the models fine-tuned above using different decoding procedures. To do this, we use a greedy decoder model and a 4-gram decoder model trained on a text corpus extracted from MediaWiki, using the same set of hyperparameters as the authors. In greater detail, we perform the decoding phase for both the BASE and LARGE versions of the model on the development clean and other validation subsets for validation, and on the 5 hours of the testing clean and other subsets for testing. In all experiments carried out below, we use the decoding algorithms presented in detail in Section 3, and state the hyperparameters that are changed.

For both the BASE and LARGE versions of the model, Baevski et al. (2020) consider three types of decoding: a greedy decoder, a 4-gram decoder model and a Transformer language model. In this report, we do not consider Transformer language models, which allow for a better understanding of the contextual dependencies in the text outputted by the model, and address

the potential of these models in Section 6. The greedy decoder contains no learning element and is thus used as described in Section 3, while the parameters for the 4-gram language model used by the authors are a beam-width of 1,500 and a weight and word insertion penalty parameters learned optimized by a hyperparameter search, such that, for the considered corpora, these parameters are equal to the values in Table 5.

Corpus	4-gram weight	4-gram word insertion
1 hour	2.90	−1.62
10 hours	2.46	−0.59
100 hours	2.15	−0.52

Table 5: Decoding hyperparameters of the 4-gram language model

In our experiments, due to time and computing limitations, we do not search for optimal hyperparameters in the decoding phase and reuse the ones displayed by Baevski et al. (2020). Additionnally, to save time during decoding, we use a beam-width of 200, which, heuristically and from early experiments on the models, has only a minor effect on the quality of the decoding using the 4-gram language model.

In the following, we display tables confronting the results we obtained to the ones published by Baevski et al. (2020). We observe that, while our results are close to the ones of the research team of wav2vec2, we fail to reproduce them exactly. we believe that these disparities are due to differences in hyperparameters not documented by the authors in the original paper, and whose setting remains ambiguous. Table 6 shows the results in terms of word error-rate (WER) obtained both by our fine-tuning approach, as well as those documented by the authors of the wav2vec2 model. Notice that each cell is a word error-rate, depending on the dataset, the version of the model (BASE or LARGE), the corpus used for fine-tuning, and different decoding strategies.

Corpus	Model	Decoder	Baevski et al. (2020)				ours			
			dev		test		dev		test	
			clean	other	clean	other	clean	other	clean	other
1 hour	BASE	Greedy	24.1	29.6	24.5	29.7	NA	NA	NA	NA
		4-gram	5.0	10.8	5.5	11.3	NA	NA	NA	NA
	LARGE	Greedy	21.6	25.3	22.1	25.3	NA	NA	NA	NA
		4-gram	4.8	8.5	5.1	9.4	NA	NA	NA	NA
10 hours	BASE	Greedy	10.9	17.4	11.1	17.6	14.2	22.5	11.7	20.3
		4-gram	3.8	9.1	4.3	9.5	9.7	10.8	2.6	12.6
	LARGE	Greedy	8.1	12.0	8.0	12.1	12.0	15.7	9.1	14.7
		4-gram	3.4	6.9	3.8	7.3	9.5	8.2	2.6	10.4
100 hours	BASE	Greedy	3.1	13.5	6.1	13.3	7.9	16.3	6.3	13.4
		4-gram	2.7	7.9	3.4	8.0	7.4	7.4	1.8	9.4
	LARGE	Greedy	4.6	9.3	4.7	9.0	5.1	11.6	4.7	9.8
		4-gram	2.3	5.7	2.8	6.0	6.6	5.6	1.6	8.4

Table 6: WER of the wav2vec2 model (BASE and LARGE versions) on multiple LibriSpeech subsets in comparison with Baevski et al. (2020)

In particular, we observe that our approach follows the results of Baevski et al. (2020) closely for large audio corpora and when the decoding is done via a greedy decoder, but remains significantly less performant for smaller corpora and for the n-gram decoding experiments. We hypothesise that this is due to differences in convergence of the pre-training phase (which have a higher impact when fine-tuning on small corpora), and on the fact that the 4-gram language models were not trained on the same text corpora. Interestingly, we also observe that, for the largest models and largest corpus, a greedy decoding strategy outperforms a 4-gram decoder, which we attribute to the fact that our language model has not seen sufficiently diverse excerpts and lacks vocabulary.

This difference between the results of Baevski et al. (2020) and ours is imputable, as highlighted above, most probably to changes in the hyperparameters of the models which are not documented exhaustively in the original paper and which were left to be set heuristically. Performing a hyper-parameter search over these is, as was noted in Section 3, not computationally feasible with limited resources and in limited time. Nonetheless, in order to better understand the dependency of these results on certain factors, we propose two analyses of our results. First, we propose a study of the WER on the reference test-other subset LibriSpeech evaluation data, indicating separately the number of insertions, deletions and substitutions, and which helps in knowing where the model fails most often. Then, we compare these results to a fine-tuning on the same sets without any pre-training to eveluate the dependence of these performances on the first, more costly phase of the training scheme.

Table 7 displays the decomposition of the WER for the test-other subset after fine-tuning the pre-trained BASE and LARGE models on different corpora. We observe that, almost always, the WER is for the largest part due to substitutions, where insertions and deletions only account for a negligible of the error. Looking at examples of transcriptions, we see that the fine-tuned wav2vec2 models often fail on one letter within a word, but get most characters right, which leads to an error in a word and an increase in WER, but which does not fully alter too much the legibility and quality of the transcription.

Corpus	Model	Decoder	WER	Insertions (I)	Deletions (D)	Substitutions (S)
1 hour	BASE	Greedy	NA	NA	NA	NA
		4-gram	NA	NA	NA	NA
	LARGE	Greedy	NA	NA	NA	NA
		4-gram	NA	NA	NA	NA
10 hours	BASE	Greedy	20.3	1.6	17.2	1.4
		4-gram	12.6	1.3	10.6	0.7
	LARGE	Greedy	14.7	1.2	12.6	0.9
		4-gram	10.4	1.0	2.9	0.5
100 hours	BASE	Greedy	13.4	1.1	11.0	1.4
		4-gram	9.4	1.0	7.8	0.6
	LARGE	Greedy	9.8	0.9	8.0	1.0
		4-gram	8.4	1.0	7.0	0.4

Table 7: Decomposition of the WER of the wav2vec2 model (BASE and LARGE versions) on the LibriSpeech test-other subset

In this regard, we also display the character error-rate (which is a less often used metric for evaluating ASR models but which is salient in revealing the flaws of underperforming models) in Table 8. In particular, we see that the CER of the wav2vec2 model is, for most models, low (*i.e.* lesser or equal than 10), and also mostly due to substitutions, and very few insertions or deletions. While these numbers remain difficult to interpret, we emphasise that they indicate that the model learns the sequential part of the task well, and that the performance issues of the model outlined above are rather linked to distinguishing between resembling phonemes.

Corpus	Model	Decoder	CER	Insertions (I)	Deletions (D)	Substitutions (S)
1 hour	BASE	Greedy	NA	NA	NA	NA
		4-gram	NA	NA	NA	NA
	LARGE	Greedy	NA	NA	NA	NA
		4-gram	NA	NA	NA	NA
10 hours	BASE	Greedy	6.7	1.6	2.8	2.4
		4-gram	5.0	1.1	2.0	1.9
	LARGE	Greedy	4.7	1.1	1.9	1.7
		4-gram	3.9	1.1	1.4	1.4
100 hours	BASE	Greedy	4.7	1.0	1.9	1.7
		4-gram	3.6	0.7	1.3	1.5
	LARGE	Greedy	3.3	0.8	1.4	1.2
		4-gram	3.1	0.7	1.2	1.2

Table 8: Decomposition of the CER of the wav2vec2 model (BASE and LARGE versions) on the LibriSpeech test-other subset

Finally, we compare the results in terms of WER between four different approaches in using pre-training weights for the fine-tuning phase. First, we compare an approach that does not leverage any pre-training phase, with the approach detailed above which does, and which uses the weights saved at the epoch at which the model provided the best accuracy for the validation set. In this regard, Table 9 shows the WER obtained after fine-tuning the models on different corpora, depending on the pre-training strategy.

Likewise, we compare an approach that uses the weights obtained at the end of the training phase, regardless of the validation set accuracy (hence prone to overfitting), with the standard approach which again uses the weights corresponding to the time when the accuracy on the development set was highest. In that regard, Table 10 shows the WER obtained after fine-tuning the models on different corpora.

We observe that the models using pre-trained weights significantly outperform the models fine-tuned from scratch, especially for the smaller corpora. With hindsight, one can say that this is a direct consequence from the self-supervised learning approach, with the fact that pre-training weights helps leverage knowledge of the phonotactic and linguistic characteristics of speech without ever using transcriptions, and fine-tuning allows to retain some of the discriminative properties of these weights. Likewise, we see that there isn't much difference between the models fine-tuned from the weights at the end of the pre-training phase or from the weights chosen to maximise the validation set accuracy, which makes sense when noting that the validation set accuracy grows almost always with time during pre-training.

Corpus	Model	Decoder	Without pre-training				With pre-training			
			dev		test		dev		test	
			clean	other	clean	other	clean	other	clean	other
1 hour	BASE	Greedy	NA	NA	NA	NA	NA	NA	NA	NA
		4-gram	NA	NA	NA	NA	NA	NA	NA	NA
	LARGE	Greedy	NA	NA	NA	NA	NA	NA	NA	NA
		4-gram	NA	NA	NA	NA	NA	NA	NA	NA
10 hours	BASE	Greedy	99.9	99.9	99.9	99.9	14.2	22.5	11.7	20.3
		4-gram	99.9	99.9	99.9	99.9	9.7	10.8	2.6	12.6
	LARGE	Greedy	99.9	99.9	99.9	99.9	12.0	15.7	9.1	14.7
		4-gram	99.9	99.9	99.9	99.9	9.5	8.2	2.6	10.4
100 hours	BASE	Greedy	99.9	99.9	99.9	99.9	7.9	16.3	6.3	13.4
		4-gram	99.9	100.0	100.0	100.0	7.4	7.4	1.8	9.4
	LARGE	Greedy	100.0	100.0	100.0	100.0	5.1	11.6	4.7	9.8
		4-gram	100.0	100.0	100.0	100.0	6.5	5.6	1.6	8.4

Table 9: WER of the wav2vec2 model (BASE and LARGE versions) on multiple LibriSpeech subsets, comparing fine-tuning from scratch or not

Corpus	Model	Decoder	Using validation criterion				Using end-of-training weights			
			dev		test		dev		test	
			clean	other	clean	other	clean	other	clean	other
1 hour	BASE	Greedy	NA	NA	NA	NA	NA	NA	NA	NA
		4-gram	NA	NA	NA	NA	NA	NA	NA	NA
	LARGE	Greedy	NA	NA	NA	NA	NA	NA	NA	NA
		4-gram	NA	NA	NA	NA	NA	NA	NA	NA
10 hours	BASE	Greedy	14.2	22.5	11.7	20.3	12.0	22.4	11.7	20.1
		4-gram	9.7	10.8	2.6	12.6	9.9	10.6	2.4	12.7
	LARGE	Greedy	12.0	15.7	9.1	14.7	10.1	15.6	9.0	14.5
		4-gram	9.5	8.2	2.6	10.4	9.1	7.8	2.4	10.1
100 hours	BASE	Greedy	7.9	16.3	6.3	13.4	6.7	16.5	6.2	13.5
		4-gram	7.4	7.4	1.8	9.4	7.6	7.2	1.8	9.8
	LARGE	Greedy	5.1	11.6	4.7	9.8	5.2	11.5	4.8	9.8
		4-gram	6.5	5.6	1.6	8.4	6.9	5.6	1.7	8.4

Table 10: WER of the wav2vec2 model (BASE and LARGE versions) on multiple LibriSpeech subsets, comparing different fine-tuning strategies

5 Adapting ASR systems across languages

In this section, we focus on the task of using self-supervised pre-training ASR models across languages, and compare multiple approaches with respect to their computational cost and performances. We focus exclusively on English and French, and use the CommonVoice corpus for French and LibriSpeech corpus for English. To do this, we leverage in great part the results obtained by (Babu et al. 2021) on the XLS-R models, which we reproduce and on which we perform experiments to guide the use of the XLS-R model in practice, as well as results documented by the literature on cross-lingual knowledge transfer and on multi-lingual speech systems.

In short, we start by giving a short overview of the different approaches to using the wav2vec2 and XLS-R models across multiple languages, as well as the computational implications they have. Then, we showcase our first approach, which consists in using mono-lingual models, *i.e.* models trained on corpora containing one language only, both for the pre-training and fine-tuning phases. In this regard, we pre-train and fine-tune wav2vec2 models on the 8th version of the French CommonVoice corpus, and compare these results to the pre-training and fine-tuning phases of the same model on the LibriSpeech corpus. Then, we consider the less computationally expensive task of fine-tuning the already pre-trained XLS-R Base and Large models, which were pre-trained on 128 languages following the procedure of Babu et al. (2021). To maintain as much similarity as possible, we fine-tune these models folowing the same training schemes as the ones used for the fine-tuning of the English and French wav2vec2 models mentionned above. Finally, we give a brief overview of the challenges that come with the pre-training phase of a multi-lingual model like XLS-R.

5.1 Universal, cross-lingual or specific modelling

As highlighted in Section 2, multiple approaches to using self-supervised learning ASR systems across multiple languages exist. One first approach can be deemed as the universal one, which works by pre-training the self-supervised deep-learning model on a wide-variety of corpora from different languages, in the hope of obtaining features that will be meaningful across different languages and with the idea of leveraging large amounts of unlabelled data. In this regard, Babu et al. (2021) pre-train a wa2vec2 architecture on corpora from 128 languages totalling 436,000 hours of speech data. In a second phase, and only in this second phase, the model can be fine-tuned on a single specific target language. One main advantage of this approach is that a single pre-training step is necessary, and the weights from this pre-training can then be used for fine-tuning (the less costly phase of training) across different languages. However, one drawback of this approach lies in the fact that the pre-training phase requires large amounts of data which are difficult to gather, and which might not provide as good results on all languages.

A second approach, typically referenced as the cross-lingual approach, refers to the idea of using a model pre-trained on a source language (*e.g.* on English) and fine-tuning it on a second, different target language *e.g.* French), in the hope that the speech features learned during the pre-training phase will be discriminative enough for the task of ASR on the target language. One advantage of this method is that the pre-training weights can be re-used, in the sense that they can be taken from another language. However, little research has adressed this method, which in practice has historically yielded relatively poor results. Heuristically, one may hypothesise that

this is due to the differences between languages, where the pre-trained model weights end up being too adapted to the source language, and, on the contrary, inadapted to the target language, especially when languages differ significantly in their phonotactic and linguistic properties.

Finally, the last approach, which is perhaps the most natural one with regards to the definition of the task, consists in using data from the target language in the two phases of training, *i.e.* pre-training the model on unlabelled data from the target language, and fine-tuning it on labelled data from the target language as well. Naturally, taking away the potential of cross-lingual transfer learning, this approach seems as the most efficient, as it leverages only the data that is useful to the end model. However, this approach is also impractical, in the sense that it requires a pre-training phase for each new language.

In the following, we present and compare two of these approaches for the case of ASR in French: the first, most naive, approach of specific modelling, using only French data in the two phases, and the second, more economical in terms of computational resources, which uses the pre-trained weights of the XLS-R model (Babu et al. 2021) as a basis before fine-tuning the language on labelled speech data in French.

5.2 Mono-lingual ASR models

Pre-training ASR models on French data

In this first, most naive mono-lingual approach for building ASR systems in French, we try to replicate the experiments of Baevski et al. (2020) and reproduced above for English, but using this time French data as our training set. To do this, we pre-train the wav2vec2 model from scratch using, apart from language-specific parameters, the same set of hyperparameters as the one used for English, hoping to obtain comparable performances. One crucial difference, however, in this endeavour, is that the 8th version of the CommonVoice corpus for French contains only roughly 600 hours of audio, compared to 960 hours in English. In more detail, we pre-train both the Base and Large versions, and use the 30 hours of the development validation subset for validation. In all experiments carried out below, we use the architecture detailed in Section 3 and state the hyperparameters that are changed.

For the Base version of the model, we use batches composed of 15.6 seconds (corresponding to 250,000 samples) cropped from the raw audio data recordings, batched together so as to not exceed 1.4 million samples per GPU. In total, on all 64 V100 GPUs, the batch-size is thus approaching 1.55 hours. The learning rate is set to follow a linear decay with a warmup consisting in the first 8% of updates, reaching a peak of 5×10^{-4}. The model is trained with a weight of $\alpha = 0.1$ for the diversity loss, with a Gumbel softmax temperature parameter annealed from 2 to a minimum of 0.5 by a factor of $1 - 5 \times 10^{-6}$ at each update, and the temperature in the contrastive loss is set to $\kappa = 0.1$. On top of this, a L2 penalty term is added to the activations of the final layer of the feature encoder to ensure regularization, the gradients a scaled down by a factor of 10, and layers are randomly dropped at each update at a rate of 5%. The number of distractors is set to 100, the number of codebooks to 2 and the number of codewords in each codebook to 320, where each of these codewords has dimension 128. The model is trained on 64 V100 GPUs for 400,000 updates.

For the LARGE version of the model, we use this time batches composed of 20.0 seconds (corresponding to 320,000 samples) cropped from the raw audio data recordings, batched together so as to not exceed 1.2 million samples per GPU. In total, on all 128 V100 GPUs, the batch-size is thus 2.67 hours. The learning rate is set to follow a linear decay with a warmup consisting in the first 8% of updates, reaching a peak of 5×10^{-5}. The model is trained with a weight of $\alpha = 0.1$ for the diversity loss, with a Gumbel softmax temperature parameter annealed from 2 to a minimum of 0.5 by a factor of $1 - 5 \times 10^{-6}$ at each update, and the temperature in the contrastive loss is set to $\kappa = 0.1$. On top of this, a dropout of 0.1 is used in the Transformer part of the model at the output of the CNN feature encoder, and layers are randomly dropped at each update at a rate of 20%. The number of distractors is set to 100, the number of codebooks to 2 and the number of codewords in each codebook to 320, where each of these codewords has dimension 384. The model is trained on 128 V100 GPUs for 250,000 updates.

To compare them to the Figures of Section 4, we display in the following figures and tables containing the same results as the one displayed for the English wav2vec2 pre-training scheme. We observe a less stable convergence in our models towards an accuracy of 60% to 70%, both on the training and validation datasets. Figure 16 displays the accuracy of the pre-training version of the BASE wav2vec2 model during training.

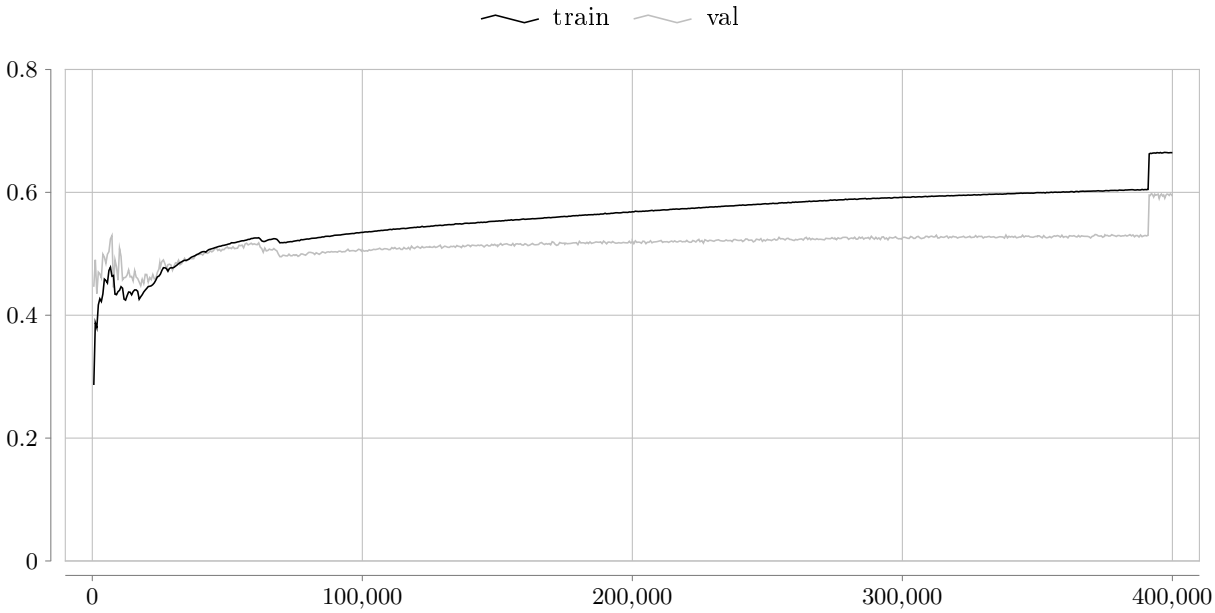


Figure 16: Accuracy of the BASE version of the wav2vec2 model during pre-training on the CommonVoice (FR) dataset

As Figure 16 shows, large jumps (steep increases or decreases in the metric's value) can be observed roughly around steps 6,000 and 70,000. Interestingly, in comparison to what is observed on the English wav2vec2 model trained on the LibriSpeech dataset, the model requires for French data much more updates to recover from these jumps, which, hypothetically, can be imputed to a more diverse dataset. As Figure 17 displays, the same relationship can be observed with code perplexity, which records jumps at the same steps of the pre-training phase.

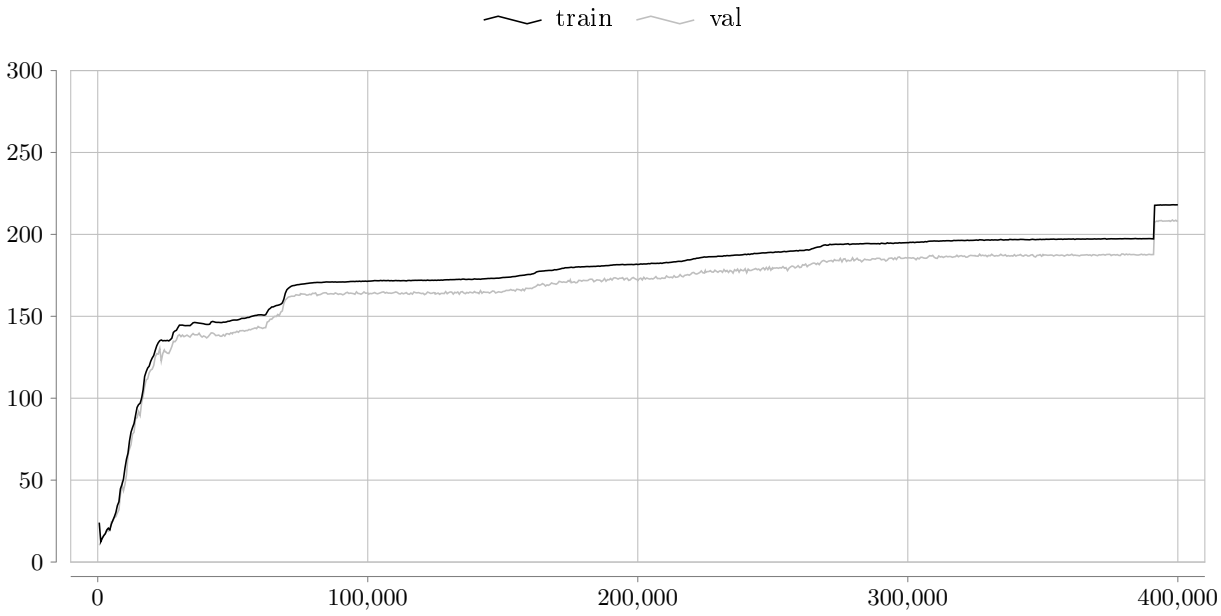


Figure 17: Code perplexity of the BASE version of the wav2vec2 model during pre-training on the CommonVoice (FR) dataset

In a similar fashion, Figure 18 displays the accuracy of the pre-training version of the LARGE wav2vec2 model during training, which reaches a peak of 62.0% at the end of training phase. The same discontiuous increase in accuracy can be observed as in the BASE version of the model.

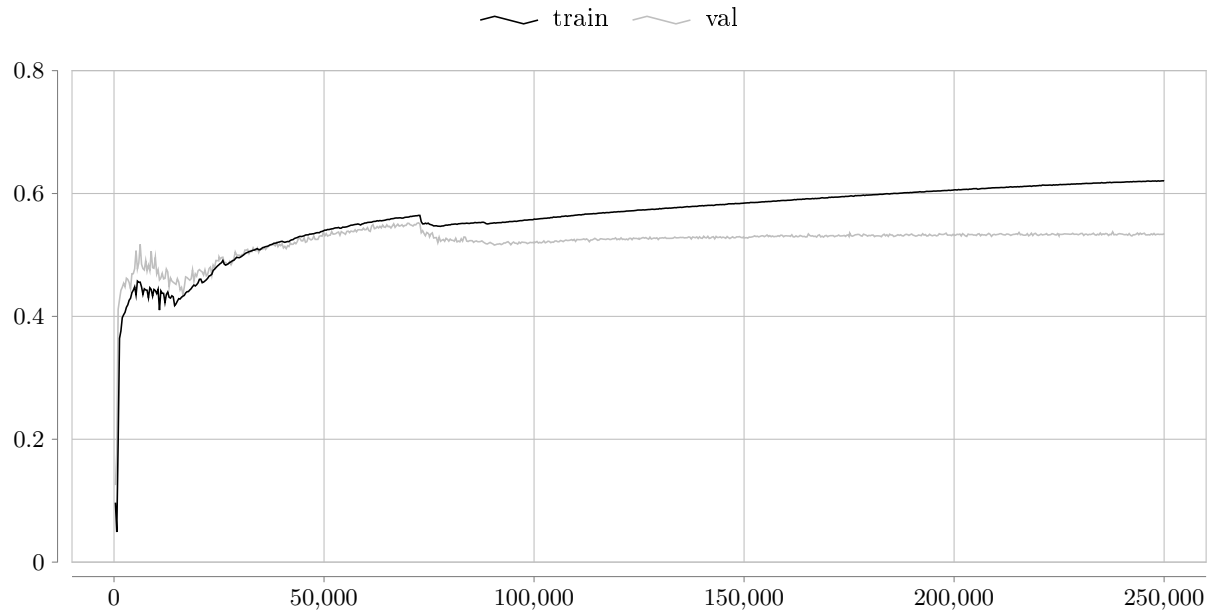


Figure 18: Accuracy of the LARGE version of the wav2vec2 model during pre-training on the CommonVoice (FR) dataset

Again, heuristically, looking at the pre-training metrics, one can hypothesise that the same mechanic is at work: *i.e.* these small jumps are due to different threshold effects on the diversity loss, where these would be linked to the annealing scheme used for the Gumbel-softmax temperature parameter. As Figure 19 shows, the code perplexity observes again the same two jumps during pre-training.

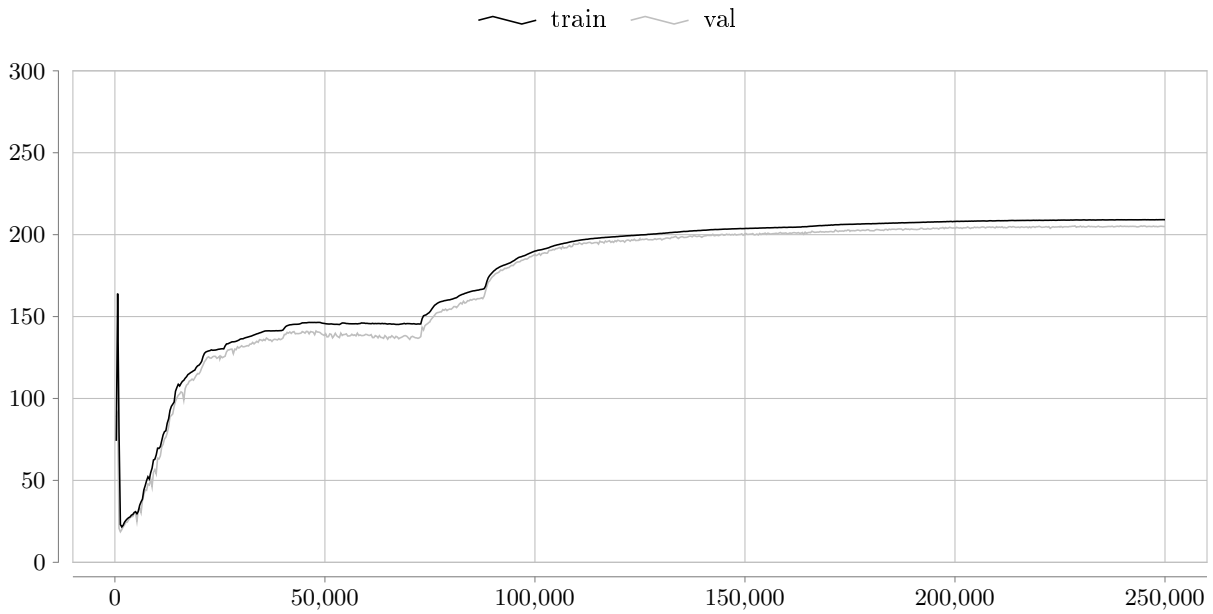


Figure 19: Code perplexity of the LARGE version of the wav2vec2 model during pre-training on the CommonVoice (FR) dataset

A summary of these results is displayed in Table 11, which mentions the results obtained for the English wav2vec2 model as well, and which indicates, both for the BASE and LARGE versions of the model, the final accuracy obtained at the end of the pre-training phase, as well as code perplexity. We see that the accuracy is slightly higher for the LARGE version of the model, but the significance of this difference remains, like in the case of the English model, open to discussion. The same can conclusion can be made for the code perplexity. However, we observe that the validation sets for the French corpus attain lower performances than their English counterparts, which we attribute to a more important domain shift.

Corpus	Version	Accuracy		Code perplexity	
		train	dev	train	dev
LibriSpeech	BASE	67.6%	73.1%	254.6	246.3
	LARGE	67.6%	72.3%	199.6	194.1
CommonVoice (FR)	BASE	66.5%	59.4%	218.0	208.3
	LARGE	62.0%	53.3%	209.1	205.0

Table 11: Final accuracy and code perplexity of the wav2vec2 English and French models during the pre-training phase

Fine-tuning mono-lingual ASR models

Following this preparatory pre-training phase, we go on by trying to replicate the results obtained on English data for the wav2vec2 model for French data from the CommonVoice French corpus by performing the fine-tuning phase. To do this, apart from language-specific parameters, and in an attempt to compare approaches, we fine-tune the wav2vec2 model using the same set of hyperparameters as used for English data. In greater detail, we perform fine-tuning for both the BASE and LARGE versions of the model on randomly selected subsets of the CommonVoice training corpus consisting in respectively 1, 10 and 100 hours of speech from audiobooks recordings, using the 30 hours of the development clean and other validation subsets for validation, and the 30 hours of the testing clean and other subsets for evaluation. In all experiments carried out below, we use the architecture detailed in Section 3 and state the hyperparameters.

For the BASE version of the model, we fine-tune the model using batches built from the raw audio extracts and composed of a maximum total of 3,200,000 samples per GPU. In total, on all 8 V100 GPUs, the batch-size is thus approaching 0.44 hours. The learning rate follows a tri-state schedule where it is first warmed-up for the first 10% of updates, then held constant for 40% of updates, and finally linearly decayed for the last 50% of updates. Its peak value, as well as the layer drop probability, time-step masking probability, channel-masking probability and number of total updates depend on the size of the corpus considered, and follow the same pattern as the one detailed in Table 3 for English data. For the LARGE version of the model, we fine-tune the model using batches built from the raw audio extracts and composed of a maximum total of 1,280,000 samples per GPU. In total, on all 24 V100 GPUs, the batch-size is thus approaching 0.53 hours. The learning rate follows the same tri-state schedule as for the BASE version of the model, and other parameters can be found in 4.

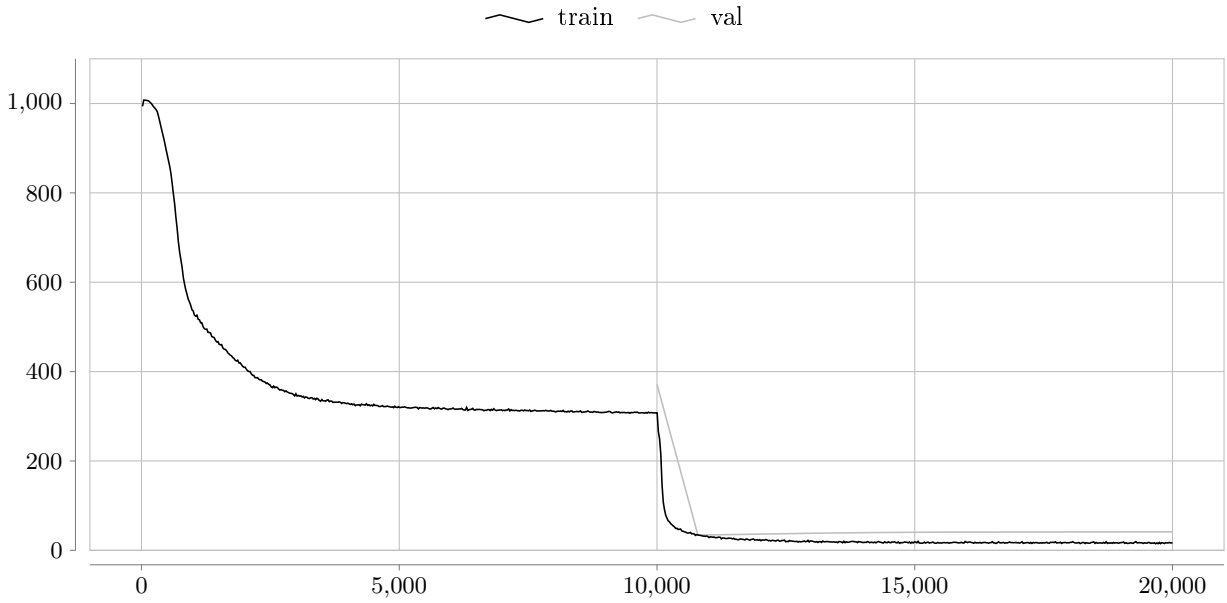


Figure 20: CTC loss of the BASE version of the wav2vec2 model during fine-training on the CommonVoice (FR) 10h sub-dataset

We first display the CTC loss functions obtained after training the models on the 10 hours corpus of the CommonVoice French dataset, using the training scheme described in Section 3. Figure 20 displays the loss of the fine-tuning version of the BASE wav2vec2 model during training on the 10h subset of the CommonVoice (FR) training data, and can be compared to Figure 14 in Section 4, which displays the same procedure on English data.

Likewise, the loss of the of the LARGE version of the wav2vec2 model during fine-tuning is displayed in Figure 21, which can be compared to Figure 21 of Section 4. These two losses show the same phases of convergence as the ones of the wav2vec2 model trained on the LibriSpeech corpus, and we do not comment further on them.

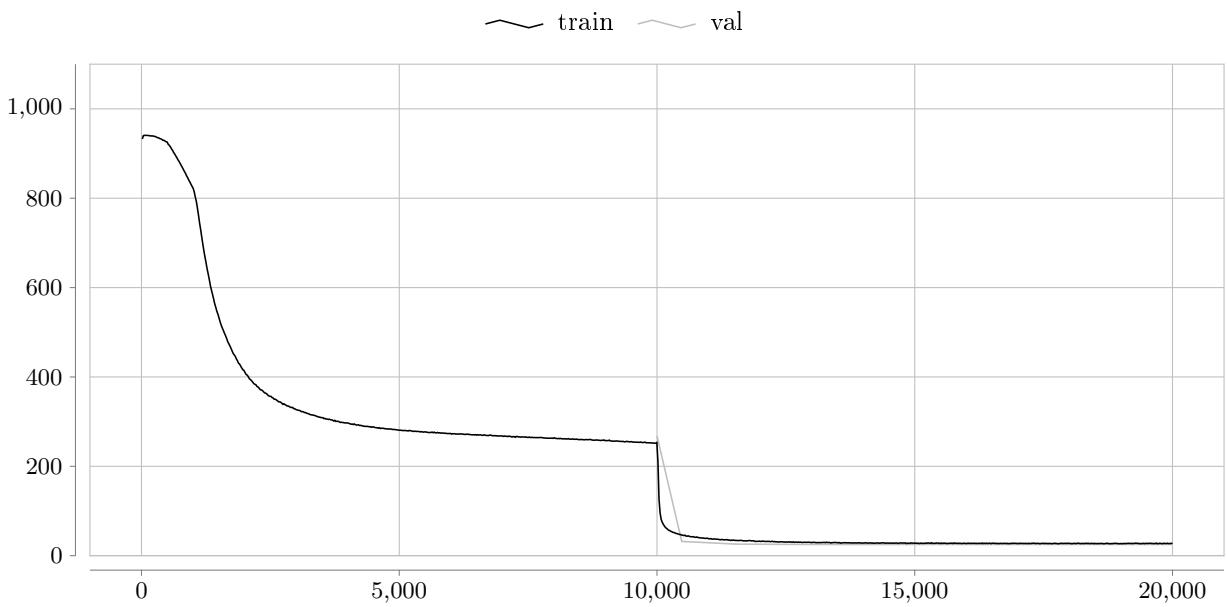


Figure 21: CTC loss of the LARGE version of the wav2vec2 model during fine-training on the CommonVoice 10h sub-dataset

For completeness, the convergence of the 1 hour and 100 hours subsets of the CommonVoice (FR) corpus follow, roughly, the same pattern as the one observed above for the 10 hours corpus. The figures displaying these losses are placed in the appendix.

Language modelling and decoding

Finally, we evaluate the models fine-tuned above using different decoding procedures. To do this, like in the setting with English data, we use a greedy decoder model and a 4-gram decoder model trained on a text corpus extracted from MediaWiki (in French), using the same set of hyperparameters as with the English language models.

We perform the decoding phase for both the BASE and LARGE versions of the model on the development clean and other validation subsets for validation, and on the 30 hours of the testing clean and other subsets for testing. The greedy decoder contains no learning element and is thus used as described in Section 3, while the parameters for the 4-gram language model

used are taken to be equal to the case of the language model in English, which undoubtedly hinders the performances of this decoding approach, but which was the only feasible option in our case. Table 12 displays the results we obtained for the wav2vec2 model trained on the French audio corpus, compared to the ones obtained for English data. We observe that the models for English significantly ouperform the models for French, which we attribute to less optimal hyper-parameters, and smaller and more diverse training sets.

Corpus	Model	Decoder	English (LS)				French (CV)	
			dev		test		dev	test
			clean	other	clean	other		
1 hour	BASE	Greedy	NA	NA	NA	NA	NA	NA
		4-gram	NA	NA	NA	NA	NA	NA
	LARGE	Greedy	NA	NA	NA	NA	NA	NA
		4-gram	NA	NA	NA	NA	NA	NA
10 hours	BASE	Greedy	14.2	22.5	11.7	20.3	33.9	46.7
		4-gram	9.7	10.8	2.6	12.6	17.4	30.0
	LARGE	Greedy	12.0	15.7	9.1	14.7	30.4	39.2
		4-gram	9.5	8.2	2.6	10.4	14.3	25.4
100 hours	BASE	Greedy	7.9	16.3	6.3	13.4	24.4	36.2
		4-gram	7.4	7.4	1.8	9.4	13.5	23.5
	LARGE	Greedy	5.1	11.6	4.7	9.8	21.1	29.4
		4-gram	6.6	5.6	1.6	8.4	11.5	19.7

Table 12: WER of the wav2vec2 model (BASE and LARGE versions) on multiple LibriSpeech and CommonVoice (FR) subsets

For reference, Table 13 displays the decomposition of the WER for the test subset after fine-tuning the pre-trained BASE and LARGE models on different corpora.

Corpus	Model	Decoder	WER	Insertions (I)	Substitutions (S)	Deletions (D)
1 hour	BASE	Greedy	NA	NA	NA	NA
		4-gram	NA	NA	NA	NA
	LARGE	Greedy	NA	NA	NA	NA
		4-gram	NA	NA	NA	NA
10 hours	BASE	Greedy	46.7	3.1	39.6	4.0
		4-gram	30.0	1.8	22.9	5.2
	LARGE	Greedy	39.2	2.2	34.0	3.0
		4-gram	25.4	1.1	18.8	5.5
100 hours	BASE	Greedy	36.2	2.3	30.5	3.4
		4-gram	23.5	1.7	17.8	4.0
	LARGE	Greedy	29.4	1.5	25.0	2.9
		4-gram	19.7	1.1	15.2	3.5

Table 13: Decomposition of the WER of the wav2vec2 model (BASE and LARGE versions) on the CommonVoice (FR) test subset

5.3 Fine-tuning multi-lingual ASR models

As underlined in Section 2, multiple works have emphasised that using multi-lingual models outperform mono-lingual ones, often claiming that this increase in performance is due to the learning of more diverse and robust speech features during pre-training (on multiple languages).

In the same fasion as in the past experiments, we first display the CTC loss functions obtained after training the models a the 10 hours corpus of the CommonVoice (FR) dataset, using the training scheme described in Section 3. Figure 22 displays the loss of the fine-tuning version of the Base XLS-R model during training on the 10h subset of CommonVoice (FR).

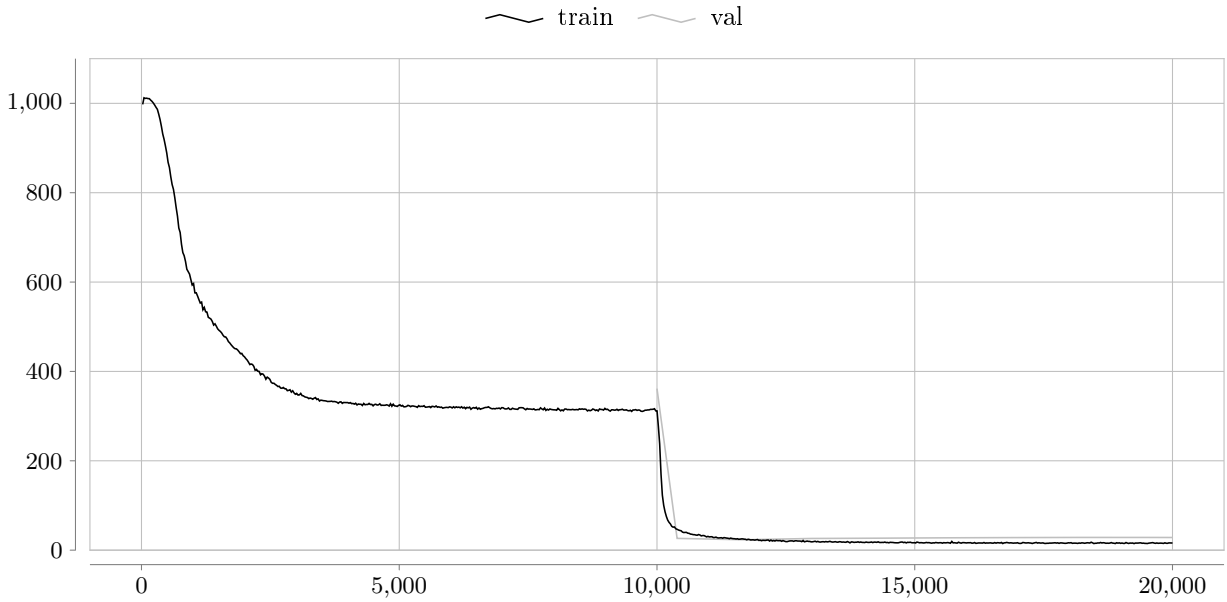


Figure 22: CTC loss of the Base version of the XLS-R model during fine-training on the CommonVoice (FR) 10h sub-dataset

Likewise, the loss of the of the Large version of the XLS-R model during fine-tuning is displayed in Figure 23. These two losses follow, roughly, the same phases of convergence as the ones of the wav2vec2 model trained on the CommonVoice corpus.

For completeness, the convergence of the 1 hour and 100 hours subsets of the CommonVoice (FR) corpus follow, roughly, the same pattern as the one observed above for the 10 hours corpus. As for the previous experiments, figures displaying these losses are placed in the appendix.

In the same fasion as with the mono-lingual approach, we evaluate the models fine-tuned above using different decoding procedures. In short, we use a greedy decoder model and a 4-gram decoder model trained on a text corpus extracted from MediaWiki (in French). We perform the decoding phase for both the Base and Large versions of the model on the development clean and other validation subsets for validation, and on the 30 hours of the testing subset for testing. Parameters for the 4-gram language model used are inhereited from the case of the language model in French. Table 14 gives a comparison of the results obtained on the wav2vec2 and XLS-R models, both fine-tuned on the same corpora.

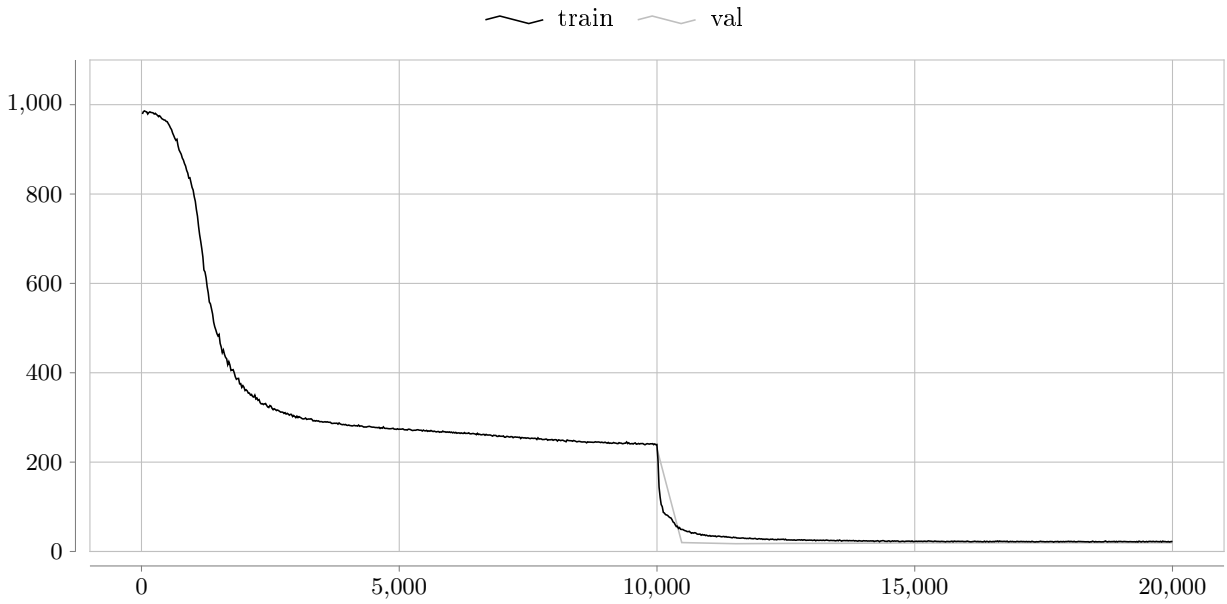


Figure 23: CTC loss of the LARGE version of the XLS-R model during fine-training on the CommonVoice 10h sub-dataset

We observe that, as highlighted by Babu et al. (2021), the XLS-R model significantly outperforms the wav2vec2 model on all corpora and for all decoding strategies. Yet, we observe that for greedy decoding, the gap is narrow, particularly given then fact that the pre-training was carried out on only 600 hours, compared to 436,000 hours on multiple languages for XLS-R.

Corpus	Model	Decoder	wav2vec2		XLS-R	
			dev	test	dev	test
1 hour	BASE	Greedy	NA	NA	NA	NA
		4-gram	NA	NA	NA	NA
	LARGE	Greedy	NA	NA	NA	NA
		4-gram	NA	NA	NA	NA
10 hours	BASE	Greedy	33.9	46.7	24.6	32.5
		4-gram	17.4	30.0	13.9	21.9
	LARGE	Greedy	30.4	39.2	20.0	26.6
		4-gram	14.3	25.4	9.7	17.9
100 hours	BASE	Greedy	24.4	36.2	17.7	25.5
		4-gram	13.5	23.5	10.3	17.6
	LARGE	Greedy	21.1	29.4	15.8	21.4
		4-gram	11.5	19.7	8.4	15.7

Table 14: WER of the XLS-R model (BASE and LARGE versions) on the testing and development CommonVoice (FR) subsets

For reference, Table 15 below displays the decomposition of the WER for the test subset after fine-tuning the pre-trained BASE and LARGE models on different corpora.

Corpus	Model	Decoder	WER	Insertions (I)	Substitutions (S)	Deletions (D)
1 hour	BASE	Greedy	NA	NA	NA	NA
		4-gram	NA	NA	NA	NA
	LARGE	Greedy	NA	NA	NA	NA
		4-gram	NA	NA	NA	NA
10 hours	BASE	Greedy	32.5	2.2	27.7	2.6
		4-gram	21.9	1.7	17.3	2.9
	LARGE	Greedy	26.6	1.6	22.6	2.4
		4-gram	17.9	1.4	13.5	3.0
100 hours	BASE	Greedy	25.5	1.5	22.0	2.0
		4-gram	17.6	1.1	14.5	2.0
	LARGE	Greedy	21.4	1.1	18.4	1.9
		4-gram	15.7	1.3	11.8	2.6

Table 15: Decomposition of the WER of the XLS-R model (BASE and LARGE versions) on the CommonVoice (FR) test subset

Finally, for completeness, we also fine-tune the XLS-R model on the LibriSpeech corpora, and compare the performance of the multi-lingual model to our pre-trained and fine-tuned wav2vec2 models, which used only 960 hours of unlabelled data in pre-training. We observe in Table 16 that the performance of the two models is mostly similar, although the wav2vec2 model is less stable for the larger corpora.

Corpus	Model	Decoder	XLS-R				wav2vec2			
			dev		test		dev		test	
			clean	other	clean	other	clean	other	clean	other
1 hour	BASE	Greedy	NA	NA	NA	NA	NA	NA	NA	NA
		4-gram	NA	NA	NA	NA	NA	NA	NA	NA
	LARGE	Greedy	NA	NA	NA	NA	NA	NA	NA	NA
		4-gram	NA	NA	NA	NA	NA	NA	NA	NA
10 hours	BASE	Greedy	7.5	12.5	6.9	12.1	14.2	22.5	11.7	20.3
		4-gram	6.9	5.0	1.4	9.7	9.7	10.8	2.6	12.6
	LARGE	Greedy	6.5	11.9	5.1	9.1	12.0	15.7	9.1	14.7
		4-gram	5.7	4.4	1.2	7.7	9.5	8.2	2.6	10.4
100 hours	BASE	Greedy	5.3	12.3	4.3	8.7	7.9	16.3	6.3	13.4
		4-gram	3.9	5.8	1.5	7.9	7.4	7.4	1.8	9.4
	LARGE	Greedy	4.3	10.2	3.5	6.9	5.1	11.6	4.7	9.8
		4-gram	5.0	3.4	1.1	6.9	6.6	5.6	1.6	8.4

Table 16: WER of the XLS-R model (BASE and LARGE versions) on the different testing and development subsets of the LibriSpeech corpus

For reference, Table 17 below displays the decomposition of the WER for the test-other subset after fine-tuning the pre-trained BASE and LARGE models on different corpora.

Corpus	Model	Decoder	WER	Insertions (I)	Substitutions (S)	Deletions (D)
1 hour	Base	Greedy	NA	NA	NA	NA
		4-gram	NA	NA	NA	NA
	Large	Greedy	NA	NA	NA	NA
		4-gram	NA	NA	NA	NA
10 hours	Base	Greedy	12.1	1.0	10.1	0.9
		4-gram	9.7	1.1	8.2	0.4
	Large	Greedy	9.1	0.7	7.7	0.8
		4-gram	7.7	0.7	6.5	0.5
100 hours	Base	Greedy	8.7	0.4	3.3	0.6
		4-gram	7.9	0.9	6.7	0.4
	Large	Greedy	6.9	0.5	5.6	0.8
		4-gram	6.9	0.8	5.6	0.5

Table 17: Decomposition of the WER of the XLS-R model (Base and Large versions) on the LibriSpeech test-other subset

5.4 Pre-training multi-lingual ASR models

Following these results, it appears quite clearly that the XLS-R models outperforms our initial approach of fine-tuning the model using weights that were pre-trained on unlabelled French data only. As highlighted in Section 6, this performance can not only be imputed to the fact that XLS-R learns features that are shared across languages and hence may represent better the space of sounds that human voices can produce, but also directly to the fact that XLS-R is pre-trained on a much higher amount of unlabelled data. This section provides a very short overview of the challenges that come with pre-training such multi-lingual models, which we highlight are both scientifically and technically (or computationally) challenging learning tasks.

As highlighted by Babu et al. (2021), one the one hand, pre-training of the different versions of the XLS-R model (Base, Large and X-Large) represents a scientific challenge. As a matter of fact, the unlabelled data from 128 languages used in pre-training comes in very diverse sets, some of which include non pre-processed speech and, most importantly, in different amounts depending on whether the considered languages are high- or low-resource. Acknowledging these potential issues, Babu et al. (2021) detail a specific pre-training procedure in which batches are composed of a fixed number languages, and where batches are randomly sampled in these languages according to a probability distribution that depends on the number of hours of training data for each language.

On the other hand, pre-training of the XLS-R model also posits a technical challenge, which dramatically hinders its reproducibility properties. In this report, the choice of using the pre-trained weights provided by the authors falls directly back to these technical issues, both in terms of collecting and storing the datasets used for training, as well as in terms of computation power. As a matter of fact, as the models are trained on 200 V100 GPUs for 1,000,000 steps, we estimate that the pre-training procedure of XLS-R is almost five times as demanding in terms of computational resources as a single pre-training of the wav2vec2 Large model on the French CommonVoice corpus.

6 Discussion and limitations

Experiments with the wav2vec2 and XLS-R models highlight two key difficulties. First, pre-training and fine-tuning wav2vec2 and XLS-R models requires large computational resources and are complex and costly procedures, which, to the best of our current ability, cannot yet benefit from large hyper-parameter optimization schemes like would be available for smaller models. Then, adaptation of these models across languages is a difficult task both on the scientific and technical levels, as these include making non-neutral choices regarding the pre-training phase (*e.g.* whether to use mono- or multi-lingual approaches), which cannot be tested other than heuristically on a small set of varying parameters, with the ability to evaluate the quality of the model only once a fine-tuning phase has been carried out.

In this section, we interpret the results obtained through the lens of these scientific and technical difficulties, and comment on the limitations of our approach as well as suggest ways in which improvements to this work could be made. In particular, we briefly mention the questions that the results presented in the experimental part of this report cannot answer, as well as ways in which these could be answered; highlight the difficulties that come with the decoding phase; and provide a short discussion on the questions that can be adressed following this work.

6.1 Results and limitations

The main results of this report are twofold. First, through the fine-tuning of a wav2vec2 model pre-trained on unlabelled French speech data, and a XLS-R model pre-trained on 128 languages from different corpora on the same labelled data in French, this report shows that mono-lingual approaches are, heuristically, not necessarily better-performing than their multi-lingual counterparts. As a matter of fact, on the data and models at hand, the XLS-R model significantly outperforms the wav2vec2 model on French data, despite the fact that the wav2vec2 model was trained exclusively on French unlabelled data.

However, this results does not fully address the question of the hierarchy of approaches between mono-lingual and multi-lingual models. As a matter of fact, while these models have been trained using the same set of hyper-parameters for evaluation purposes, comparison between the two is not straightforward. First, it is likely that both of these models could be improved with a more carefully chosen set of hyper-parameters, and the question of which model would be more performant using these sets of parameters remains unanswered. Then, the approaches are not fully comparable in the amount of data used: the XLS-R model uses a large corpus consisting in more than 436,000 hours of unlabelled speech data, whilst the wav2vec2 model used only about 600 hours of unlabelled data during pre-training. The difference between these approaches thus remains to be explored in its full extent. Nonetheless, our experiments highlight that, on a practical level, the use of pre-trained multi-lingual models is not associated with a decrease in performance when compared to the pre-training of a model on a single, specific language.

Then, comparing the fine-tuning phase across multiple labelled datasets of different lengths in French and in English, this report shows that although self-supervised approaches leverage unlabelled data in the sense that they improve the performance of the models when compared to a fine-tuning from scratch, they are still highly dependent on the amount of labelled data available

for the fine-tuning phase. In this regard, we show that, with the data and models at hand, a corpus of 10 hours is necessary to obtain results of sufficient quality for interpretation, and a corpus of 100 hours allows to improve by a fair margin these performances. However, yet again, these results can only be interpreted on a practical level, as a thorough hyper-parameter search was not conducted for smaller corpora, thus the full potential of these models for fine-tuning on limited resources cannot be observed.

6.2 Decoding challenges

As highlighted in Section 4, decoding is a highly important phase in the training of a performant ASR system. Further, as was shown in the experimental part of this report, different methods for decoding lead to different performances, where more advanced approaches such as decoding with a n-gram language model sometimes lead to wide gaps in terms of WER when compared to more basic approaches such as greedy decoding.

In this regard, multiple recent works, including Baevski et al. (2020) leverage the better generalization properties of attention-based language models (Vaswani et al. 2017), which lead to state-of-the-art performances on common corpora for the task of ASR. In this work, these decoding language models are not adressed, as they constitute themselves an open research question which requires vast amounts of work to be understood in their full extent. Importantly, these models allow for a better taking into account of out-of-vocabulary words, as they allow to distinguish between words that are plausibly in a language vocabulary, and words that are not, rather than working in a rule-based way as do n-gram language models.

More importantly, as with the fine-tuning and pre-training phases, the decoding approaches used in this model did not leverage the full potential of the models in the sense that no search for hyper-parameters was carried out, and the weights used by Baevski et al. (2020) were used instead, although the language models were different. In that sense, it is worth noting that, to some extent, results in terms of WER using n-gram language model decoding could be improved, particularly on the French datasets. Moreover, training of a n-gram language models requires access to large corpora of text data, which constitute a barrier for the use of these ASR systems on low-resource languages.

6.3 Further work

Following these results, one can suggest two main directions as ways forward to continue this research. First, and most straightforwardly, one could be drawn towards the idea of replicating these experiences on multiple languages, including ones that are different, both on the phonotactic and linguistic levels, from French and English, and to pursue research in the direction of finding optimized hyper-parameters for the tasks of pre-training and fine-tuning.

Then, as pointed out in Section 3, another line of work may consist in research on domain adaptation. As a matter of fact, all corpora used in this report come from clean data, recorded either for the task of ASR or for the making of audiobooks. Yet, many ASR systems are being used on noisy data, which does not come from the same domain as the data on which wav2vec2 and XLS-R were pre-trained. In this line, work towards making these ASR systems performant on data from other domains, such as GSM phone conversations, is much needed.

7 Conclusion

ASR systems have known impressive improvements in the last few decades, which can, for the most part, be traced back to a long history of research on speech processing and in the use of deep-learning algorithms for tasks that inherently leverage sequential data. Yet, as highlighted by this report, even the most recent of these improvements fail to adress some of the major bottlenecks in the field: the question of multi-lingual ASR on the first hand, and the question of domain adaptation in the other.

In this work, our first contribution is to produce a thorough guide to the use of self-supervised learning model architectures for the task of ASR, in particular the use of the wav2vec2 model of Baevski et al. (2020) and the follow-up XLS-R model of Babu et al. (2021). In this regard, we highlight the dependence of the two training phases (pre-training and fine-tuning) for the creation of an efficient ASR system, and, in particular, detail the different blocks of the two training phases which each play an important role in the learning process. Importantly, self-supervised learning posits a supplementary difficulty in the learning process when compared to supervised learning approaches, as the performance of the model can only be evaluated after the two phases have been completed, a characteristic which largely hinders the ability of research teams to study the effect of hyper-parameters on the model heuristically given the important computational cost of the pre-training phase.

Our second, and perhaps most important contribution is to study the adaptability of the wav2vec2 and XLS-R models across languages, and evaluate their performances when trained to perform ASR in French following different procedures. In that regard, we show that, with the data at hand, and with a practical point of view, multi-lingual models such as XLS-R, which were pre-trained on a corpus composed of multiple languages, significantly outperform mono-lingual models trained on the target language only, for almost all lengths of fine-tuning corpora, both on the French CommonVoice corpus and on the LibriSpeech corpus. The differences in terms of performance can, indubitably, not only be attributed to the multi-lingual properties of the pre-training phase, and are most probably due in great part to the larger amount of data used during pre-training and the carefully chosen set of hyper-parameters. Nevertheless, on a practical level, considering the important computational costs linked with the pre-training of a wav2vec2 model, especially on corpora with multiple thousands hours of speech data, this report emphasises that re-using the pre-trained weights of XLS-R provided by Babu et al. (2021) is likely to yield better results than peforming a specific pre-training. Finally, this report shows that while self-supervised learning schemes largely reduce the need of large labelled datasets for high-quality ASR, a minimal corpus size of about 10 hours of labelled, and a language model based decoding are still needed to obtain exploitable results.

Following these conclusions, it appears that while training self-supervised ASR models on multiple languages is an attainable objective for high- and medium-resource languages, these experiments largely depend on the quality of the data, which on the corpora used in most recent research are wide-band, studio-quality speech extracts. Comparably, real-life data is usually noisy, and uses lower-quality codecs. In this regard, a study of domain adaptation methods for self-supervised ASR systems is much needed.

A Appendix

Glossary

ANN Artificial Neural Network

ASR Automatic Speech Recognition

CER Character Error-Rate

CNN Convolutional Neural Network

DNN Deep Neural Network

E2E End-to-End *e.g. for a deep-learning model*

GPU Graphical Processing Unit

HMM Hidden Markov Model

LPC Linear Predictive Coding

LSTM Long Short-Term Memory Network

MLP Multi-Layer Perceptron

NA Not Available

OOV Out-of-Vocabulary *i.e. unknown grapheme for a language model*

PER Phoneme Error-Rate

RNN Recurrent Neural Network

UER Utterance Error-Rate

WER Word Error-Rate

Additional datasets and languages

This appendix section covers a wide array of datasets commonly used in ASR research, with detailed features on the amount of data available, its characteristics (labelled or not), and the presence of official data splits for training, validation and evaluation purposes. Additionally, a summary table of the correspondence between languages and datasets is displayed below, where the columns names correspond to the abbreviations of the dataset names.

Language	WS	LS	LL	TI	CH	TL	CV	GS	VP	ML	VL	BA	AS
English	■	■	■	■	■	■	■	■	■	■	■		
French							■		■	■	■		
German							■		■	■	■		
Spanish							■		■	■	■		
Italian							■		■	■	■		
Chinese							■				■	■	■
Russian							■				■		

Table 18: Availability of languages in commonly used speech corpora

Wall Street Journal (WSJ) WSJ (Garofalo et al. 2007) is a commonly used corpus for evaluating and training supervised ASR models, which consists of about 400 hours of speech from read articles of the Wall Street Journal sampled at 16kHz, along with their original transcriptions. Data subsets in the WSJ corpus are: a training set of about 360 hours, a development and testing subset of roughly 20 hours and a testing set of 20 hours too. All speech in spoken in English and was recorded in studio conditions.

LibriSpeech (LS) LibriSpeech (Panayotov et al. 2015) is a widely used corpus for speech-to-text tasks, which is part of the wider LibriVox project (see below) and which consists of about 1000 hours of speech from audiobooks recordings sampled at 16kHz along with their transcriptions. Data subsets in the LibriSpeech corpus are: a training set of roughly 960 hours, two development sets of roughly 5 hours, and two testing sets of roughly 5 hours as well. The development and testing subsets are split into: a subset designated as clean, consisting of excerpts read by speakers with low word-error rate (WER; and another subset designated as other, consisting of speech from speakers with higher WER. Additionally, to account for low-resource settings, the training set is typically divided in subsets of respectively 10 minutes, 1 hour, 10 hours and 100 hours of speech. In line with the corpus' source, all speech is spoken in English by professional readers.

Libri-light (LL) Libri-light (Kahn, Rivière, et al. 2019) is another corpus of spoken English audio files derived from the LibriVox project and designed at providing data for training ASR systems with limited or no supervision. It consists of 60,000 hours of audio from audiobooks recordings sampled at 16kHz. Since it originates from the LibriVox project, and consist of audio data with no transcriptions, Libri-light typically uses the LibriSpeech development and testing datasets as benchmark. Alike LibriSpeech, all speech is spoken in English by professional readers.

TIMIT (TI) The TIMIT (Garofolo 1993) dataset, and more specifically the acoustic-phonetic continuous speech part of the TIMIT dataset, is a commonly used corpus for ASR, which consists of about 4 hours of speech recordings of phonetically rich sentences read by 630 speakers and sampled at 16kHz. Data subsets are: a training set of about 3 hours, a development set of about half an hour and a testing set of about half and hour too. All speech is spoken in English and recorded in studio conditions.

CHiME (CH) The CHiME corpora (Barker et al. 2018) are multiple versions of the same dataset created as part of a series of challenges for advances in speech separation and speech recognition. The main aim of the challenges and henceforth datasets is to provide speech data for training robust speech recognition systems. In its fifth release, the dataset consists of roughly 50 hours of recorded conversations in real indoor environments sampled at 16kHz. Data subsets are: a training set of about 40 hours; a development set of 4.5 hours and a testing set of 5.5 hours. All speech is spoken in English and recorded in real-world environments with background noises.

TED-LIUM (TL) The TED-LIUM corpora (Hernandez et al. 2018) are another series of datasets aimed at improving performances of ASR in English. In its third release, it consists in 452 hours of transcribed speech data recorded during TED events such as TED talks and TED conferences sampled at 16kHz. Data subsets are: a training set of roughly 440 hours; a development set of about 5 hours and a testing set of 5 hours as well. All speech is spoken in English, but following the variety of speakers, the data contains strong variations in terms of recording conditions and accent.

Common Voice (CV) The CommonVoice project, run by Mozilla, is an open-source, crowd-sourced speech data gathering platform where users are asked to read out sentences in their native languages, or to assess the quality and validity of recordings made by other users. In its ninth release, CommonVoice contains about 15000 hours of recorded speech sampled at 16kHz with corresponding transcriptions in 93 languages. Data subsets are not all split in training, development and testing sets, hence altering the potential for reproducibility in publications. Speech is recorded in a wide array of languages, and with varying recording conditions although almost always in good quality.

GigaSpeech (GS) The GigaSpeech corpus (Guoguo Chen et al. 2021) is a large dataset of transcribed speech audio that aims at providing high-quality labelled audio data from varied sources to improve the robustness of data-intensive ASR systems. To do this, it covers a wide variety of speech recordings (audiobooks, podcasts, YouTube videos, etc.) on a broad range of subjects. At the time of writing, GigaSpeech contains 10,000 hours of speech data sampled at 16kHz, along with all corresponding transcriptions. Data subsets consist in: five training subsets of respectively 10, 250, 1,000, 2,500 and 10,00 hours of speech respectively; a development set of 12.5 hours and a testing set of 40.3 hours. All speech is spoken in English, and while the data consists in clean, non-overlapping speech, recordings contain some background noises and originate from different domains.

VoxPopuli (VP) The VoxPopuli corpus (Changhan Wang et al. 2021) is a large scale multi-lingual dataset aimed at improving unsupervised and semi-supervised representation learning for a variety of speech-processing tasks. In its original release, it contains about 400,000 hours of unlabelled speech data spread across 23 languages, as well as 17,300 hours of labelled speech data in 15 languages. All data is sourced from the European Parliament event recordings. Data subsets include language-wise subsets, as well training, development and testing data subsets for all languages with transcribed data where the ratio for these datasets are respectively 18/20, 1/20 and 1/20. Speech recordings are clean and contain little background noise.

Multi-lingual LibriSpeech (ML) Multi-lingual LibriSpeech (Pratap, Q. Xu, et al. 2020) is a large-scale multi-lingual corpus derived from the LibriVox project and aimed at improving the cross-lingual properties of ASR end-to-end self-supervised models. It consists in about 50,000 hours of speech recordings from audiobooks, of which 44,500 are in English and 6,000 in European languages (German, Dutch, French, Spanish, Italian, Portuguese and Polish). All speech is sampled at 16kHz and was recorded in studio conditions for published audiobooks. Data partitions include language-wise subsets of varied sizes, along with training, development and testing sets for all languages with different ratios depending on the quantity of speech available.

VoxLingua (VL) VoxLingua (Valk et al. 2020) is a large corpus of automatically collected, mostly unlabelled web speech audio data sourced from YouTube videos and aimed at helping in training automatic speech recognition models in an unsupervised way. In total, Voxlingua contains 6,628 hours of speech content scattered across 107 languages (with on average 62 hours per language). All data was resampled at 16kHz and was recorded in varied conditions depending on the content of the YouTube video from which it was extracted, hence making it useful for domain adaptation tasks. Data partitions include the unlabelled training set, along with a small evaluation set of 1609 verified utterances from 33 languages.

BABEL (BA) BABEL (Gales et al. 2014) is a widely used dataset for automatic speech recognition in multiple low-resource languages, which consists in data from conversational telephone speech communications. BABEL consists in about 1,000 hours of speech data in 17 African or Asian languages, with about 80 hours of transcribed audio data per language on average. All data is sampled at 8kHz, and was recorded during real-life telephone conversations, hence is very noisy. Data subsets include language-wise corpora, each of which contains a full language pack of about 80 hours per language, along with small language pack, used for development and of roughly 10 hours of transcribed speech.

AIShell (AS) AIShell-1 (Bu et al. 2017) and AIShell-2 (J. Du et al. 2018) are open-source speech corpora in Chinese Mandarin. AIShell-1 consists in 170 hours of recorded speech from 400 speakers with varied accents conducted in a quiet environment, in studio recording conditions and sampled at 16kHz. AIShell-2 consists in 1000 hours of clean reading speech data recorded on participants smartphones in more varied conditions, also sampled at 16kHz. Data subsets consist in training sets for both AIShell-1 and AIShell-2, along with development and testing subsets recorded by users using their smartphone. All speech is spoken in Mandarin by users with a wide variety of accents.

Additional figures and tables

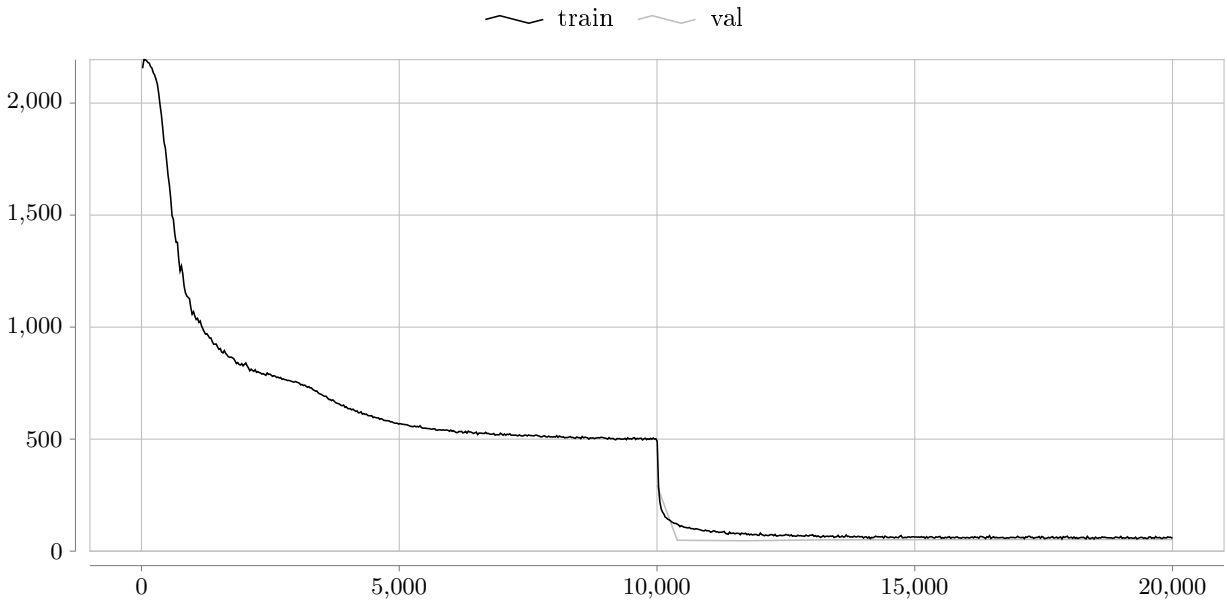


Figure 24: CTC loss of the BASE version of the wav2vec2 model during fine-tuning on the LibriSpeech 1h sub-dataset

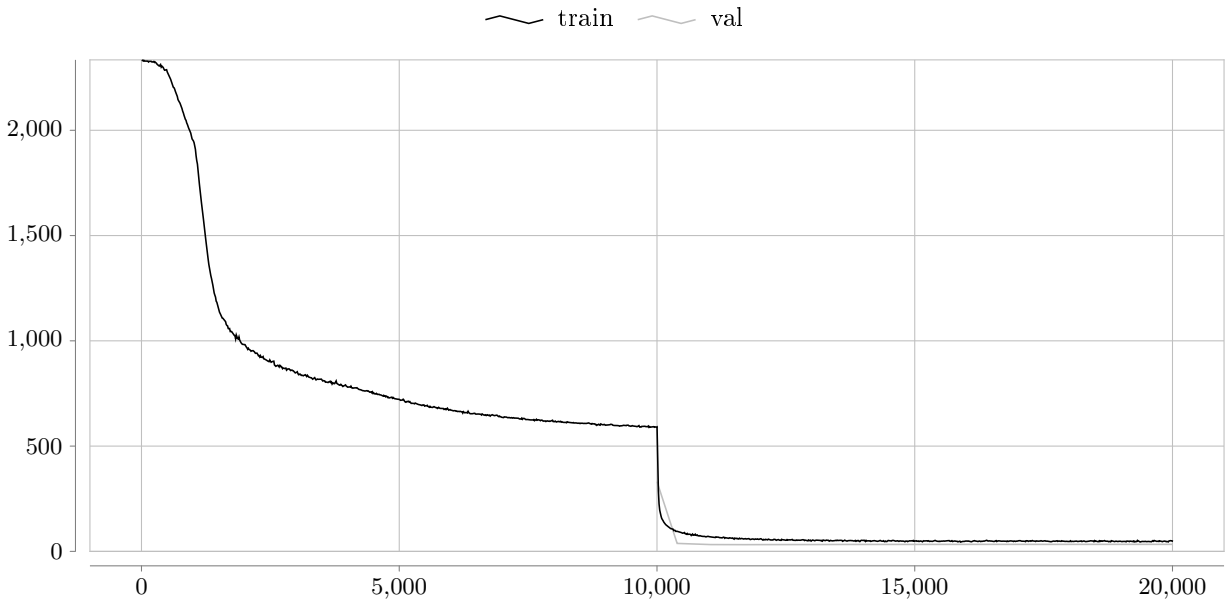


Figure 25: CTC loss of the LARGE version of the wav2vec2 model during fine-tuning on the LibriSpeech 1h sub-dataset

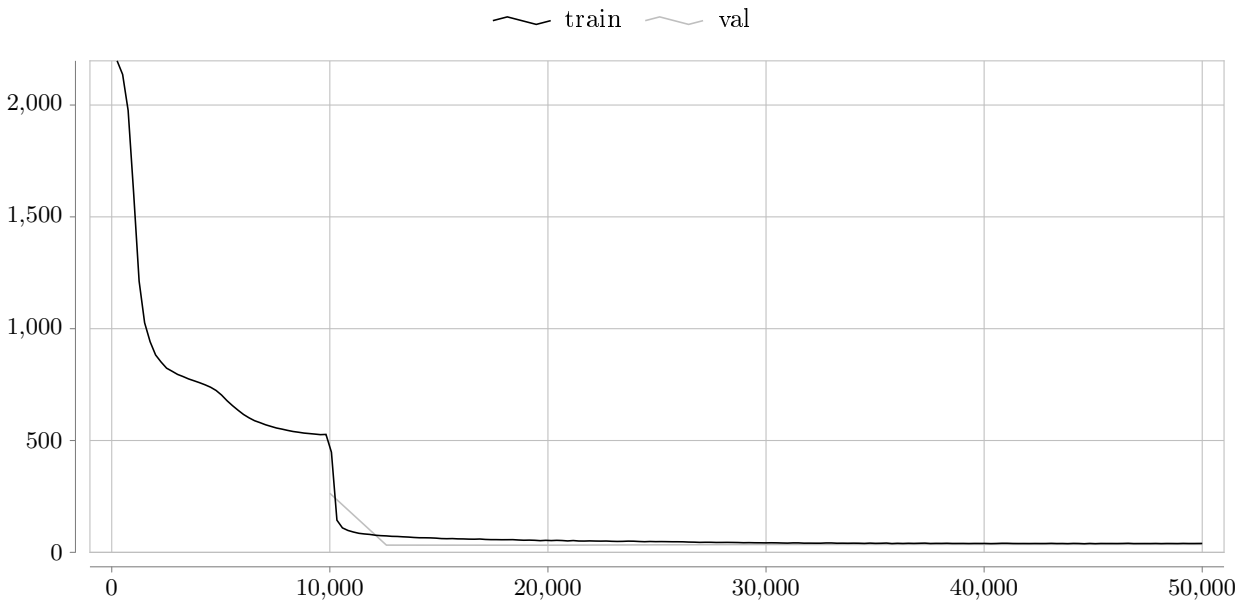


Figure 26: CTC loss of the BASE version of the wav2vec2 model during fine-tuning on the LibriSpeech 100h sub-dataset

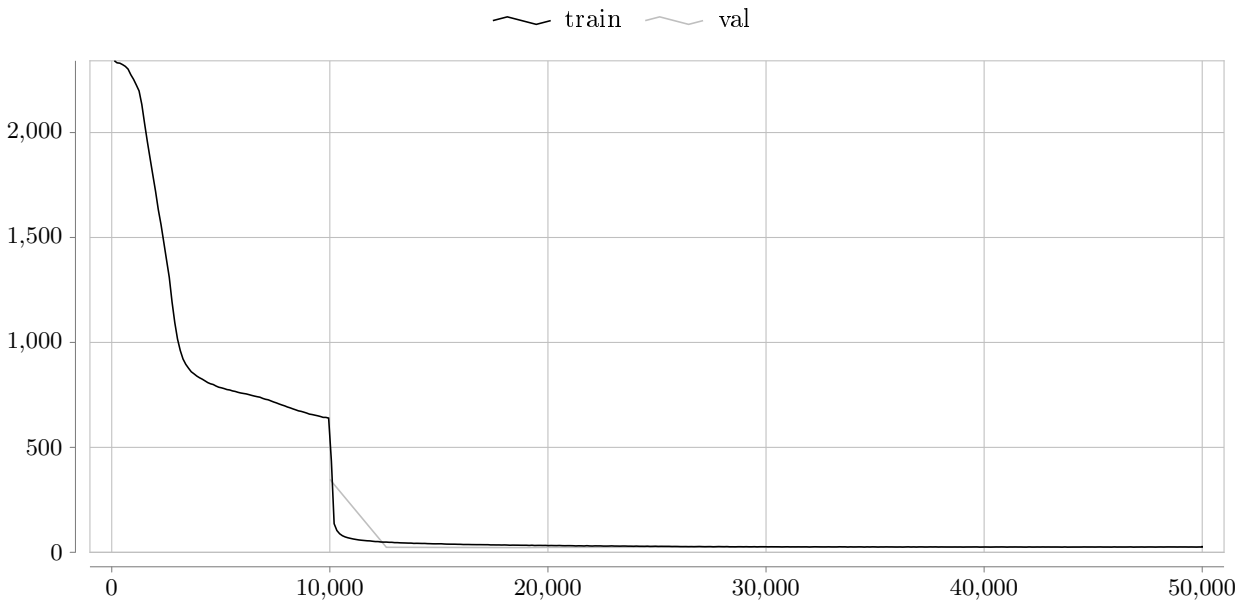


Figure 27: CTC loss of the LARGE version of the wav2vec2 model during fine-tuning on the LibriSpeech 100h sub-dataset

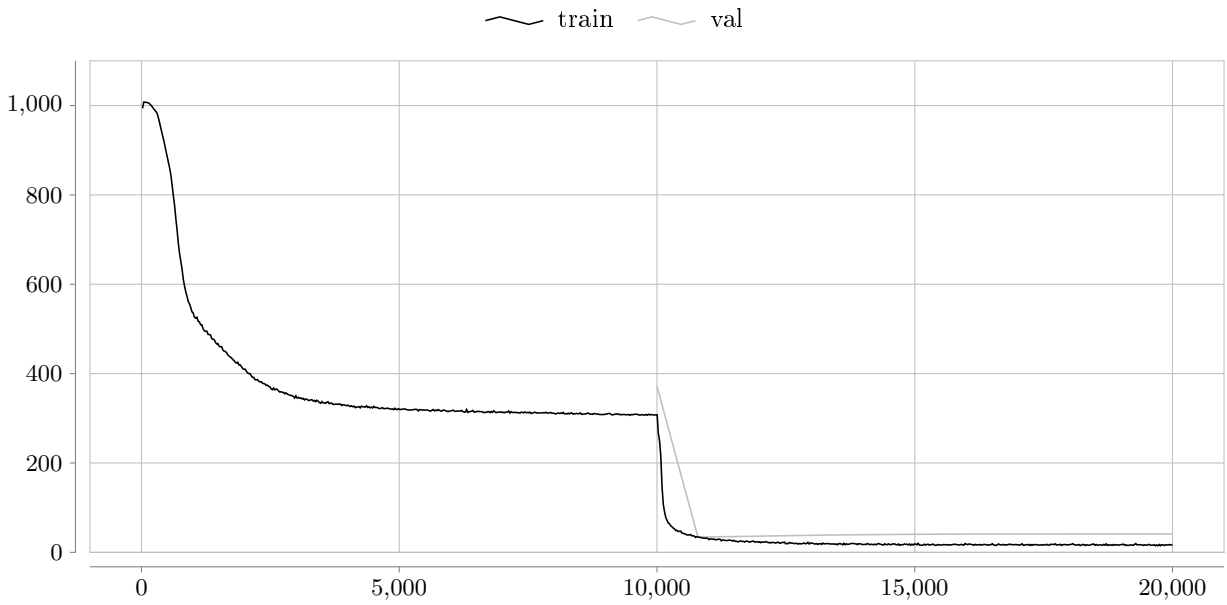


Figure 28: CTC loss of the BASE version of the wav2vec2 model during fine-tuning on the CommonVoice (FR) 1h sub-dataset

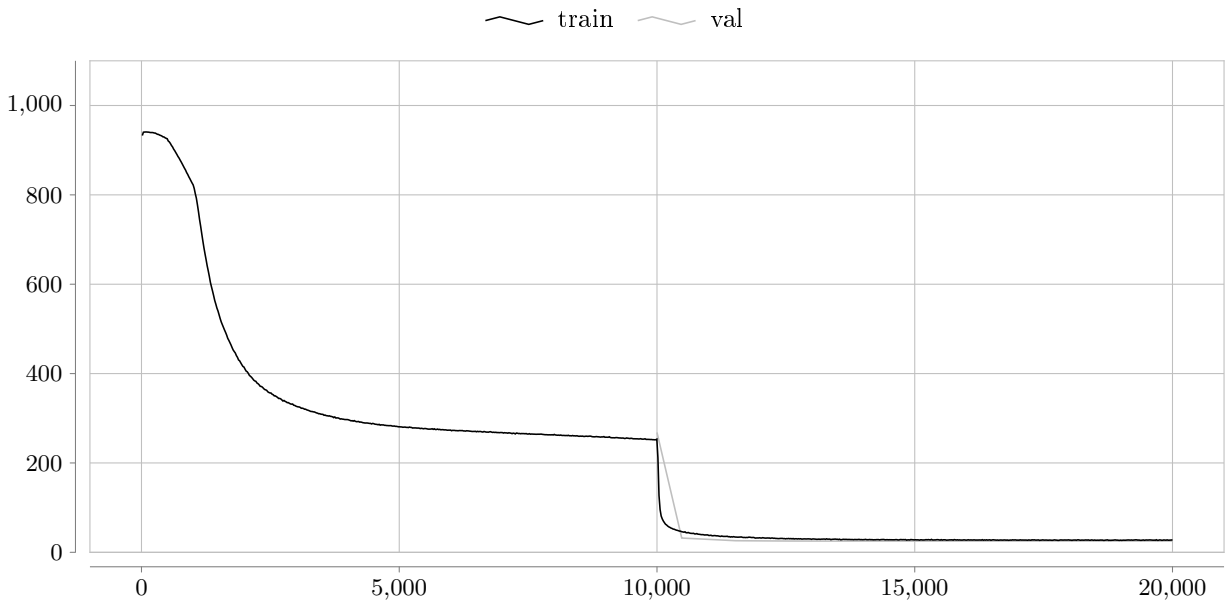


Figure 29: CTC loss of the LARGE version of the wav2vec2 model during fine-tuning on the CommonVoice (FR) 1h sub-dataset

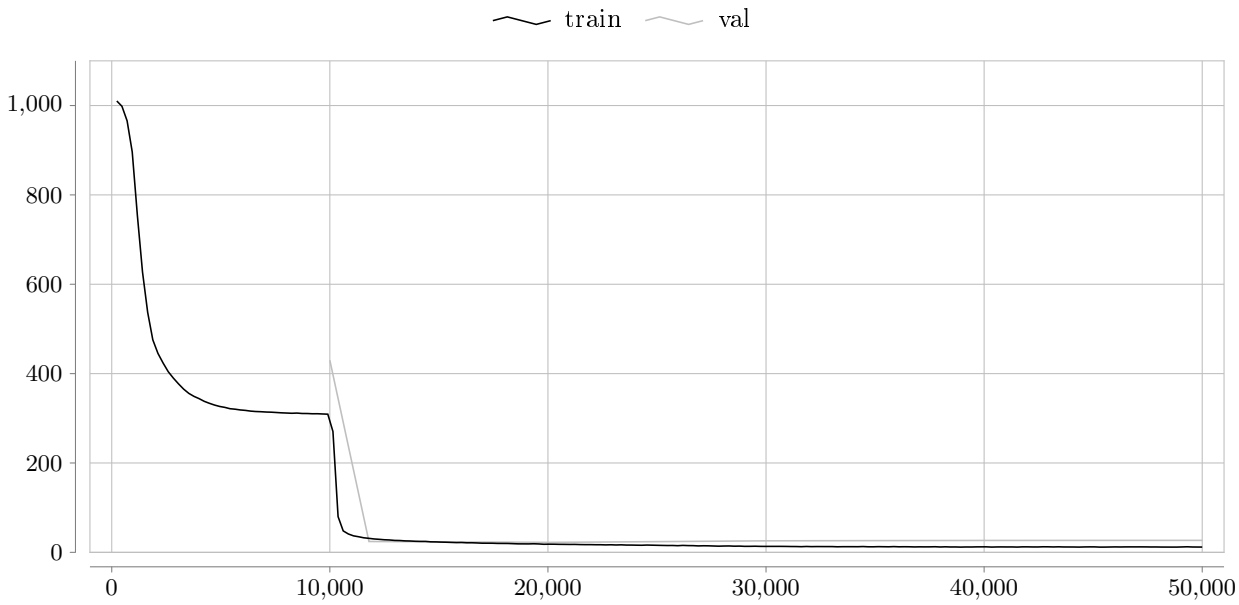


Figure 30: CTC loss of the BASE version of the wav2vec2 model during fine-tuning on the CommonVoice (FR) 100h sub-dataset

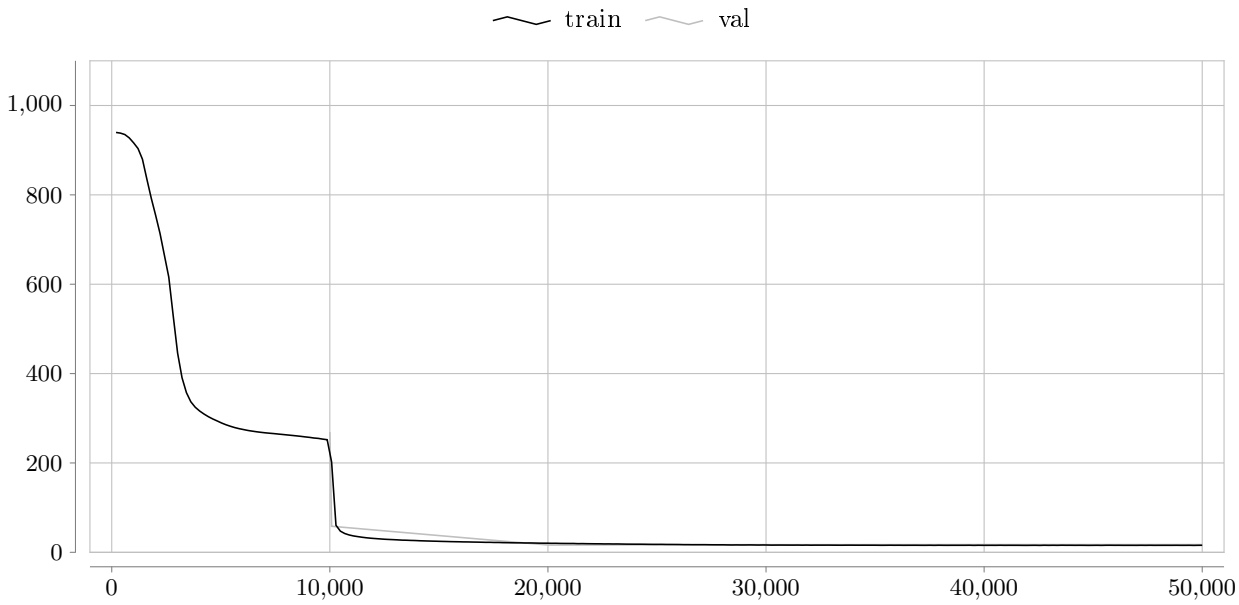


Figure 31: CTC loss of the LARGE version of the wav2vec2 model during fine-tuning on the CommonVoice (FR) 100h sub-dataset

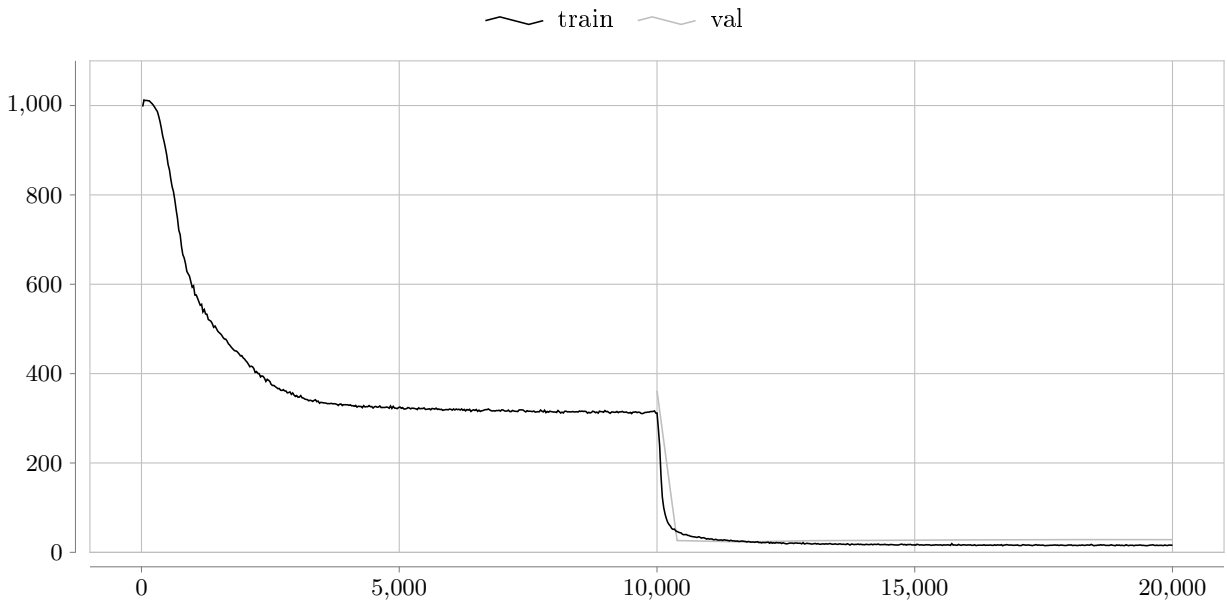


Figure 32: CTC loss of the BASE version of the XLS-R model during fine-tuning on the CommonVoice 1h sub-dataset

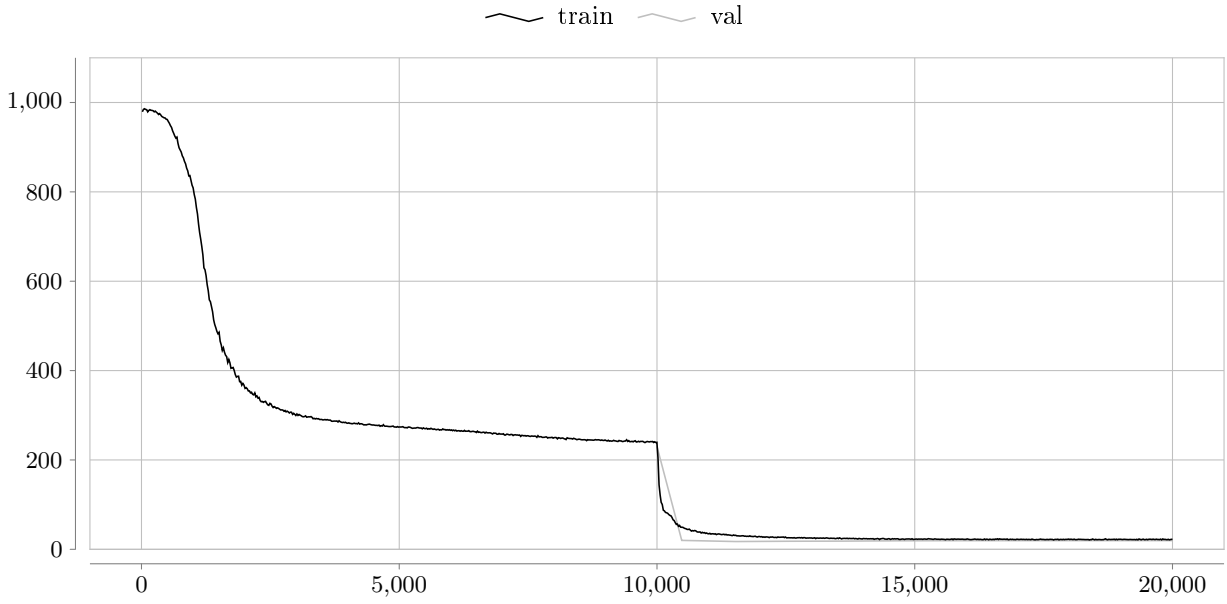


Figure 33: CTC loss of the LARGE version of the XLS-R model during fine-tuning on the CommonVoice (FR) 1h sub-dataset

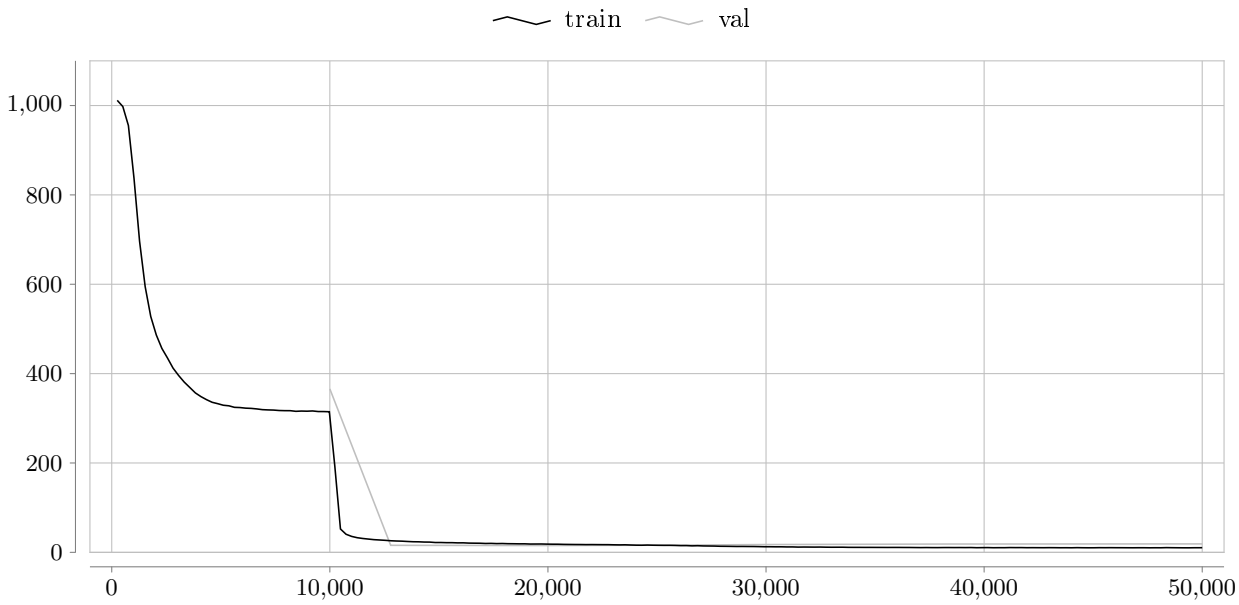


Figure 34: CTC loss of the BASE version of the XLS-R model during fine-tuning on the CommonVoice (FR) 100h sub-dataset

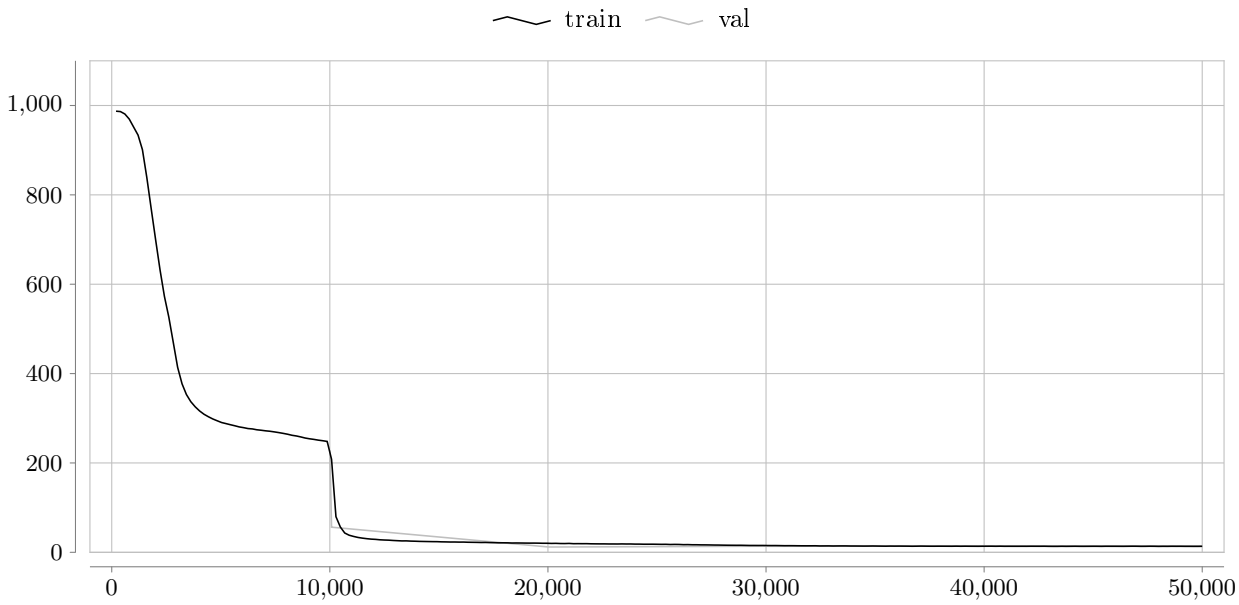


Figure 35: CTC loss of the LARGE version of the XLS-R model during fine-tuning on the CommonVoice (FR) 100h sub-dataset

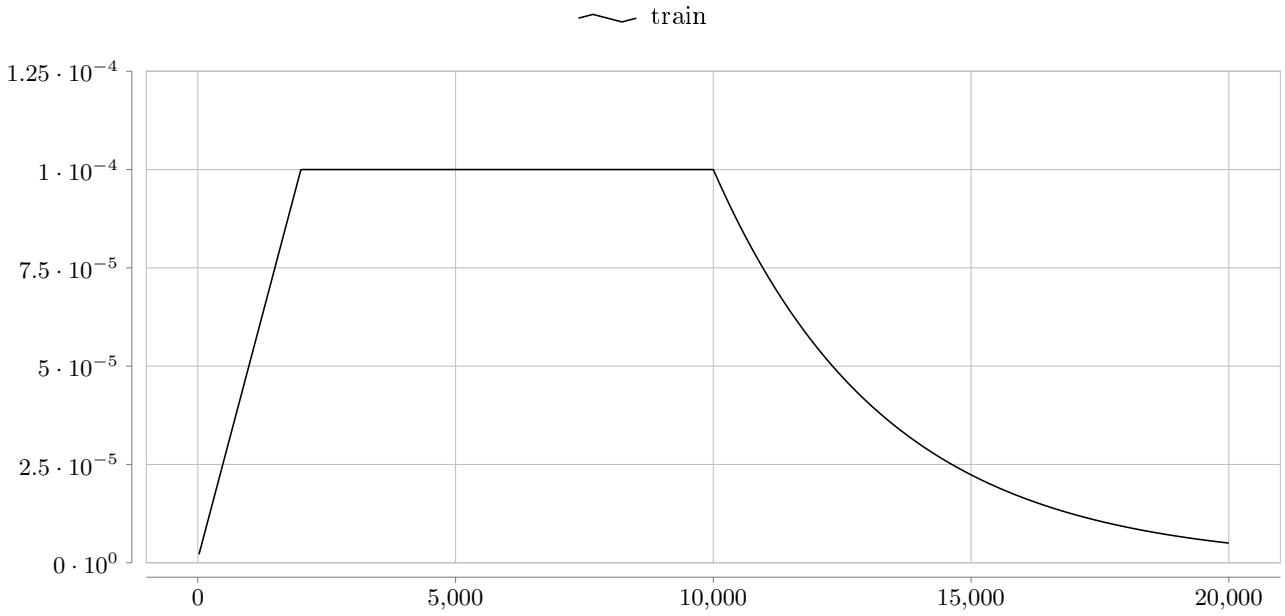


Figure 36: Learning rate schedule for the BASE version of the wav2vec model for a CommonVoice (FR) or LibriSpeech 10h sub-dataset

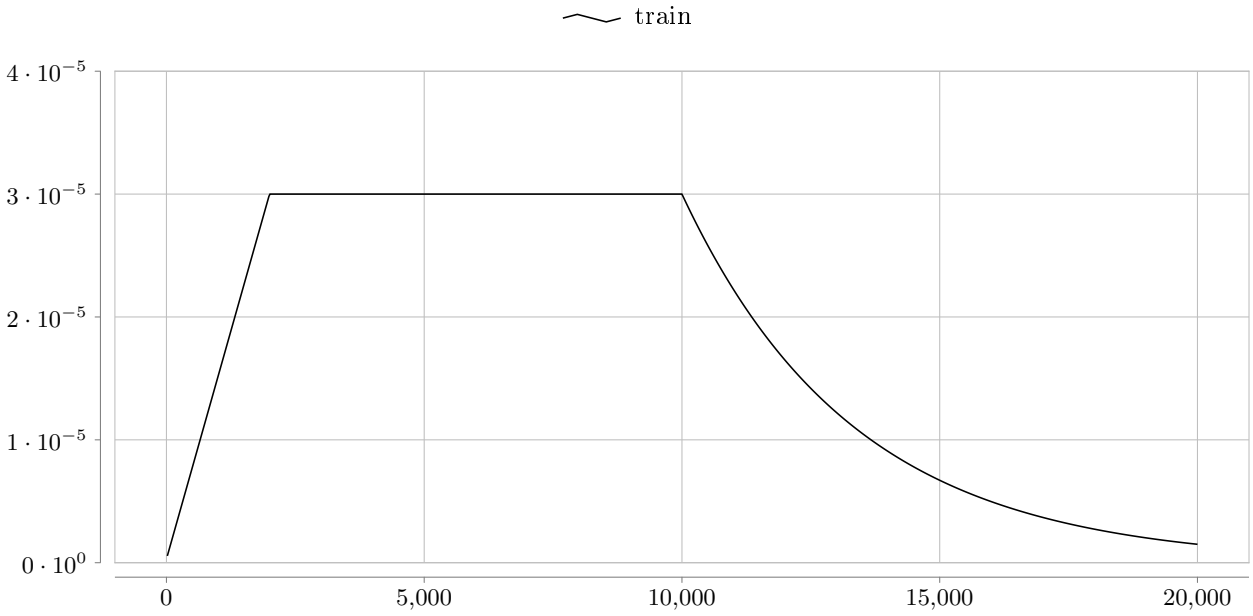


Figure 37: Learning rate schedule for the LARGE version of the wav2vec model for a CommonVoice (FR) or LibriSpeech 10h sub-dataset

Figures

1 End-to-end two-phases self-supervised model dual architecture for ASR 7
2 wav2ec pre-training model architecture 9
3 wav2ec2 pre-training model architecture 10
4 HuBeRT pre-training model architecture 13
5 WavLM pre-training model architecture 14
6 wav2ec2 pre-training model architecture 25
7 wav2ec2 fine-tuning model architecture 31
8 Example of computation of the CTC loss function 33
9 Example of posteriogram in the decoding phase 35
10 Accuracy of the Base version of the wav2vec2 model during pre-training on the LibriSpeech dataset 40
11 Code perplexity of the Base version of the wav2vec2 model during pre-training on the LibriSpeech dataset 41
12 Accuracy of the Large version of the wav2vec2 model during pre-training on the LibriSpeech dataset 41
13 Code perplexity of the Base version of the wav2vec2 model during pre-training on the LibriSpeech dataset 42
14 CTC loss of the Base version of the wav2vec2 model during fine-training on the LibriSpeech 10h sub-dataset 44
15 CTC loss of the Large version of the wav2vec2 model during fine-training on the LibriSpeech 10h sub-dataset 45
16 Accuracy of the Base version of the wav2vec2 model during pre-training on the CommonVoice (FR) dataset 52
17 Code perplexity of the Base version of the wav2vec2 model during pre-training on the CommonVoice (FR) dataset 53
18 Accuracy of the Large version of the wav2vec2 model during pre-training on the CommonVoice (FR) dataset 53
19 Code perplexity of the Large version of the wav2vec2 model during pre-training on the CommonVoice (FR) dataset 54
20 CTC loss of the Base version of the wav2vec2 model during fine-training on the CommonVoice (FR) 10h sub-dataset 55
21 CTC loss of the Large version of the wav2vec2 model during fine-training on the CommonVoice 10h sub-dataset 56
22 CTC loss of the Base version of the XLS-R model during fine-training on the CommonVoice (FR) 10h sub-dataset 58
23 CTC loss of the Large version of the XLS-R model during fine-training on the CommonVoice 10h sub-dataset 59
24 CTC loss of the Base version of the wav2vec2 model during fine-tuning on the LibriSpeech 1h sub-dataset 69
25 CTC loss of the Large version of the wav2vec2 model during fine-tuning on the LibriSpeech 1h sub-dataset 69
26 CTC loss of the Base version of the wav2vec2 model during fine-tuning on the LibriSpeech 100h sub-dataset 70

27 CTC loss of the LARGE version of the wav2vec2 model during fine-tuning on the LibriSpeech 100h sub-dataset . . . 70
28 CTC loss of the BASE version of the wav2vec2 model during fine-tuning on the CommonVoice (FR) 1h sub-dataset . . . 71
29 CTC loss of the LARGE version of the wav2vec2 model during fine-tuning on the CommonVoice (FR) 1h sub-dataset . . . 71
30 CTC loss of the BASE version of the wav2vec2 model during fine-tuning on the CommonVoice (FR) 100h sub-dataset . . . 72
31 CTC loss of the LARGE version of the wav2vec2 model during fine-tuning on the CommonVoice (FR) 100h sub-dataset . . . 72
32 CTC loss of the BASE version of the XLS-R model during fine-tuning on the CommonVoice 1h sub-dataset . . . 73
33 CTC loss of the LARGE version of the XLS-R model during fine-tuning on the CommonVoice (FR) 1h sub-dataset . . . 73
34 CTC loss of the BASE version of the XLS-R model during fine-tuning on the CommonVoice (FR) 100h sub-dataset . . . 74
35 CTC loss of the LARGE version of the XLS-R model during fine-tuning on the CommonVoice (FR) 100h sub-dataset . . . 74
36 Learning rate schedule for the BASE version of the wav2vec model for a CommonVoice (FR) or LibriSpeech 10h sub-dataset . . . 75
37 Learning rate schedule for the LARGE version of the wav2vec model for a CommonVoice (FR) or LibriSpeech 10h sub-dataset . . . 75

Tables

1 WER of recent end-to-end models on the LibriSpeech corpus . . . 15
2 Final accuracy and code perplexity of the Base and Large versions of the wav2vec2 model during the pre-training phase . . . 42
3 Fine-tuning hyperparameters of the Base version of the wav2vec2 model . . . 43
4 Fine-tuning hyperparameters of the Large version of the wav2vec2 model . . . 44
5 Decoding hyperparameters of the 4-gram language model . . . 46
6 WER of the wav2vec2 model (Base and Large versions) on multiple LibriSpeech subsets in comparison with Baevski et al. (2020) . . . 46
7 Decomposition of the WER of the wav2vec2 model (Base and Large versions) on the LibriSpeech test-other subset . . . 47
8 Decomposition of the CER of the wav2vec2 model (Base and Large versions) on the LibriSpeech test-other subset . . . 48
9 WER of the wav2vec2 model (Base and Large versions) on multiple LibriSpeech subsets, comparing fine-tuning from scratch or not . . . 49
10 WER of the wav2vec2 model (Base and Large versions) on multiple LibriSpeech subsets, comparing different fine-tuning strategies . . . 49
11 Final accuracy and code perplexity of the wav2vec2 English and French models during the pre-training phase . . . 54
12 WER of the wav2vec2 model (Base and Large versions) on multiple LibriSpeech and CommonVoice (FR) subsets . . . 57
13 Decomposition of the WER of the wav2vec2 model (Base and Large versions) on the CommonVoice (FR) test subset . . . 57
14 WER of the XLS-R model (Base and Large versions) on the testing and development CommonVoice (FR) subsets . . . 59
15 Decomposition of the WER of the XLS-R model (Base and Large versions) on the CommonVoice (FR) test subset . . . 60
16 WER of the XLS-R model (Base and Large versions) on the different testing and development subsets of the LibriSpeech corpus . . . 60
17 Decomposition of the WER of the XLS-R model (Base and Large versions) on the LibriSpeech test-other subset . . . 61
18 Availability of languages in commonly used speech corpora . . . 66

Bibliography

Amodei, Dario et al. (2016). "Deep Speech 2: End-to-End Speech Recognition in English and Mandarin". In: *International Conference on Machine Learning*. PMLR, pp. 173–182.

Atal, Bishnu S and Suzanne L Hanauer (1971). "Speech analysis and synthesis by linear prediction of the speech wave". In: *The Journal of the Acoustical society of America* 50.28, pp. 637–655.

Azeemi, Abdul Hameed, Ihsan Ayyub Qazi, and Agha Ali Raza (2022). "Towards Representative Subset Selection for Self-Supervised Speech Recognition". In: *arXiv preprint arXiv:2203.09829*.

Babu, Arun et al. (2021). "Xls-r: Self-supervised cross-lingual speech representation learning at scale". In: *arXiv preprint arXiv:2111.09296*.

Baevski, Alexei et al. (2020). "wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations". In: *arXiv preprint arXiv:2006.11477*.

Baker, James (1975). "The DRAGON system – An overview". In: *IEEE Transactions on acoustics, speech and signal processing* 23.1, pp. 24–29.

Barker, Jon et al. (2018). "The fifth CHiME speech separation and recognition challenge: dataset, task and baseline". In: *arXiv preprint arXiv:1803.10609*.

Baskar, Murali Karthick et al. (2022). "Ask2Mask: Guided Data Selection for Masked Speech Modeling". In: *arXiv preprint arXiv:2202.12719*.

Baum, Leonard E et al. (1972). "An inequality and associated maximisation technique in statistical estimation for probabilistic functions of Markov processes". In: *Inequalities* 3.1, pp. 1–8.

Baum, Leonard E and ted Petrie (1966). "Statistical inference for probabilistic functions of finite state Markov chains". In: *The annals of mathematical statistics* 37.6, pp. 1554–1563.

Bellman, Richard (1954). "The theory of dynamic programming". In: *Bulletin of the American Mathematical Society* 60.6, pp. 503–515.

Bello, Irwan et al. (2019). "Attention Augmented Convolutional Networks". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3286–3295.

Bourlard, Hervé A and Nelson Morgan (2012). *Connectionist Speech Recognition: A Hybrid Approach*. Vol. 247. Springer Science & Business Media.

Bu, Hui et al. (2017). "AIShell-1: An Open-Source Mandarin Speech Corpus and A Speech Recognition Baseline". In: *Oriental COCOSDA 2017*, Submitted.

Chan, William, Navdeep Jaitly, et al. (2015). "Listen, attend and spell". In: *arXiv preprint arXiv:1508.01211*.

Chan, William, Daniel Park, et al. (2021). "Speechstew: Simply mix all available speech recognition data to train one large neural network". In: *arXiv preprint arXiv:2104.02133*.

Chen, Guoguo et al. (2021). "Gigaspeech: An evolving, multi-domain asr corpus with 10,000 hours of transcribed audio". In: *arXiv preprint arXiv:2106.06909*.

Chen, Sanyuan et al. (2021). "Wavlm: Large-scale self-supervised pre-training for full stack speech processing". In: *arXiv preprint arXiv:2110.13900*.

Chen, Li-Wei and Alexander Rudnicky (2021). "Exploring Wav2vec 2.0 fine-tuning for improved speech emotion recognition". In: *arXiv preprint arXiv:2110.06309*.

Chi, Zewen et al. (2021). "XLM-E: Cross-lingual Language Model Pre-training via ELECTRA". In: *arXiv preprint arXiv:2106.16138*.

Chung, Yu-An, Wei-Hung Weng, et al. (2018). "Unsupervised cross-modal alignment of speech and text embedding spaces". In: *Advances in Neural Information Processing* 31.

Chung, Yu-An, Yu Zhang, et al. (2021). "W2v-bert: Combining contrastive learning and masked language modeling for self-supervised speech pre-training". In: *arXiv preprint arXiv:2108.06209*.

Collobert, Ronan, Awni Hannun, and Gabriel Synnaeve (2019). "A fully differentiable beam-search decoder". In: *International Conference on Machine-Learning*. PMLR, pp. 1341–1350.

Conneau, Alexis et al. (2020). "Unsupervised Cross-lingual Representation Learning for Speech Recognition". In: *arXiv preprint arXiv:2006.13979*.

Cuervo, Santiago et al. (2021). "Contrastive prediction strategies for unsupervised segmentation and categorization of phonemes and words". In: *arXiv preprint arXiv:2110.15909*.

Davis, Ken H, R Biddulph, and Stephen Balakesh (1952). "Automatic recognition of spoken digits". In: *The Journal of the Acoustical Society of America* 24.6, pp. 637–642.

Denes, P (1959). "The design and operation of the mechanical speech recogniser at University College London". In: *Journal of the British Institution of radio Engineers* 19.4, pp. 219–229.

Devlin, Jacob et al. (2018). "BeRT: Pre-training of Deep Bi-directional Transformers for Language Understanding". In: *arXiv preprint arXiv:1810.04805*.

Doersch, Carl, Abhinav Gupta, and Alexei A Efros (2015). "Unsupervised visual representation learning by context prediction". In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1422–1430.

Du, Jiayu et al. (2018). "AIShell-2: Transforming Mandarin ASR Research Into Industrial Scale". In: *arXiv preprint arXiv:1808.10583*.

Elman, Jeffrey L and David Zipser (1988). "Learning the hidden structure of speech". In: *The Journal of the Acoustical Society of America* 83.4, pp. 1615–1626.

Eyben, Florian et al. (2009). "From speech to letters - Using a novel neural network architecture for grapheme based ASR". In: *2009 IEEE Workshop on Automatic Speech Recognition and Understaning*. IEEE, pp. 376–380.

Ferguson, JD (1980). "Hidden Markov analysis: an introduction". In: *Hidden Markov Models for Speech*.

Forgie, James W and Carma D Forgie (1959). "Results obtained from a vowel recognition computer program". In: *The Journal of the Acoustical Society of America* 31.11, pp. 1480–1489.

Gales, Mark JF et al. (2014). "Speech Recognition and Keyword Spotting for Low-resource Languages: BABEL Project Research at CUED". In: *Fourth International Workshop on Spoken Language Technologies for Under-Resourced languages (SLTU-2014)*. International Speech Communication Association (ISCA), pp. 16–23.

Ganin, Yaroslav and Victor Lempitsky (2014). "Unsupervised Domain Adaptation by Backpropagation". In: *arXiv preprint arXiv:1409.7495*.

Ganin, Yaroslav, Evgeniya Ustinova, et al. (2016). "Domain-adversarial Training of Neural Networks". In: *The Journal of Machine Learning Research* 17.1, pp. 2096–2030.

Garofalo, John et al. (2007). "CSR-I WSJ Complete". In: *Linguistic Data Consortium, Philadelphia*.

Garofolo, John S (1993). "Timit acoustic phonetic continuous speech corpus". In: *Linguistic Data Consortium, 1993*.

Gondi, Santosh (2022). "Wav2Vec2. 0 on the Edge: Performance Evaluation". In: *arXiv preprint arXiv:2202.05993*.

Graves, Alex, Santiago Fernàndez, et al. (2006). "Connectionnist temporal classification: labelling unsegmented sequence data with recurrent neural networks". In: pp. 369–376.

Graves, Alex and Navdeep Jaitly (2014). "Towards end-to-end speech recognition with recurrent neural networks". In: *International Conference on Machine Learning*. PMLR, pp. 1764–1772.

Gulati, Anmol et al. (2020). "Conformer: Convolution-augmented transformer for speech recognition". In: *arXiv preprint arXiv:2005.08100*.

Gumbel, Emil Julius (1954). *Statistical theory of extreme values and some practical applications: a series of lectures*. Vol. 33. U.S. Government Printing Office.

Hannun, Awni S et al. (2014). "First-pass Large Vocabulary Continuous Speech Recognition Using Bi-Directional Recurrent DNNs". In: *arXiv preprint arXiv:1408.2873*.

Hannun, Awni et al. (2014). "Deep Speech: Scaling Up End-to-End Speech Recognition". In: *arXiv preprint arXiv:1412.5567*.

He, Hao, Kaiwen Zha, and Dina Katabi (2022). "Indiscriminate Poisoning Attacks on Unsupervised Contrastive Learning". In: *arXiv preprint arXiv:2202.11202*.

Heigold, Georg et al. (2013). "Multilingual Acoustic Models using Distributed Deep Neural Networks". In: *IEEE Internation Conference on Acoustics, Speech and Signal Processing*. IEEE, pp. 8619–8623.

Hernandez, François et al. (2018). "TED-LIUM 3: twice as much data and corpus repartition for experiments on speaker adaptation". In: *arXiv preprint arXiv:1805.04699*.

Hinton, Geoffrey et al. (2012). "Deep neural networks for acoustic modeing in speech recognition: The shared views of four research groups". In: *IEEE Signal processing magazine* 29.6, pp. 82–97.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long short-term memory". In: *Neural computation* 9.8, pp. 1735–1780.

Hsu, Wei-Ning, Benjamin Bolte, et al. (2021). "HuBERT: Self-Supervised Speech Representation Learning by Masked Prediction of Hidden Units". In: *arXiv preprint arXiv:2106.07447*.

Hsu, Wei-Ning, Anuroop Sriram, et al. (2021). "Robust wav2vec 2.0: Analyzing domain shift in self-supervised pre-training". In: *arXiv preprint arXiv:2104.01027*.

Huang, Jiayuan et al. (2006). "Correcting sample selection bias by unlabeled data". In: *Advances in neural information processing systems* 19.

Huang, Jui-Ting et al. (2013). "Cross-lingual knowledge transfer using multilingual deep neural network with shared hidden layers". In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, pp. 7304–7308.

Hwang, Dongseong, Ananya Misra, et al. (2021). "Large-scale ASR Domain Adaptation using Self-and Semi-supervised Learning". In: *arXiv preprint arXiv:2110.00165*.

Hwang, Dongseong, Khe Chai Sim, et al. (2022). "Pseudo Label Is Better Than Human Label". In: *arXiv preprint arXiv:2203.12668*.

Ikatura, Fumitada (1970). "A statistical method for estimation of speech spectral density and formant frequencies". In: *Electronics and Communications in Japan* 53.1, pp. 36–43.

— (1975). "Minimum prediction residual principle applied to speech recognition". In: *IEEE Transactions on acoustics, speech, and signal processing* 23.1, pp. 67–72.

Jang, Eric, Shixing Gu, and Ben Poole (2016). "Categorical Reparametrization with Gumbel-softmax". In: *arXiv preprint arXiv:1611.01144*.

Jelinek, Frederick, Lalit Bahl, and Robert Mercer (1975). "Design of a linguistic statistical decoder for the recognition of continuous speech". In: *IEEE Transactions on Information Theory* 21.3, pp. 250–256.

Jiang, Dongwei et al. (2019). "Improving Transformer-based Speech Recognition using Unsupervised Pre-training". In: *arXiv preprint arXiv:1910.09932*.

Joshi, Pratik et al. (2020). "The State and Fate of Linguistic Diversity and Inclusion in the NLP World". In: *arXiv preprint arXiv:2004.09095*.

Joshi, Raviraj and Anupam Singh (2022). "A Simple Baseline for Domain Adaptation in End-to-End ASR Systems using Synthetic Data". In: *Proceedings of the Fifth Workshop on E-Commerce and NLP*, pp. 244–249.

Kahn, Jacob, Ann Lee, and Awni Hannun (2019). "Self-Training for End-to-End Speech Recognition". In: *arXiv preprint arXiv:1909.09116*.

Kahn, Jacob, Morgane Rivière, et al. (2019). "Libri-Light: A Benchmark for ASR with Limited or No Supervision". In: *arXiv preprint arXiv:1912.07875*.

Kamper, Herman (2022). "Word Segmentation on Discovered Phone Units with Dynamic Programming and Self-Supervised Scoring". In: *arXiv preprint arXiv:2202.11929*.

Kanda, Naoyuki et al. (2022). "Streaming Multi-Talker ASR with Token-Level Serialized Output Training". In: *arXiv preprint arXiv:2202.00842*.

Khurana, Sameer et al. (2020). "Unsupervised Domain Adaptation for Speech Recognition via Uncertainty Driven Self-Training". In: *arXiv preprint arXiv:2011.13439*.

Kingma, Diederik P and Jimmy Ba (2014). "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980*.

Kong, Zhifeng et al. (2022). "Speech Denoising in the Waveform Domain with Self-Attention". In: *arXiv preprint arXiv:2202.07790*.

Kubo, Yotaro, Shigeki Karita, and Michiel Bacchiani (2022). "Knowledge Transfer from Large-scale Pretrained Language Models to End-to-end Speech Recognizers". In: *arXiv preprint arXiv:2202.07894*.

Kuznetsova, Anastasia et al. (2022). "Curriculum optimization for low-resource speech recognition". In: *arXiv preprint arXiv:2202.08883*.

Levinson, Stephen E, Lawrence R Rabiner, and M Mohan Sondhi (1983). "An introduction to the application of the theory of probabilistic functions of a Markov process to automatic speech recognition". In: *Bell System Technical Journal* 62.4, pp. 1035–1074.

Li, Jinyu (2021). "Recent Advances in End-to-End Automatic Speech Recognition". In: *arXiv preprint arXiv:2111.01690*.

Li, Mohan et al. (2022). "Transformer-based Streaming ASR with Cumulative Attention". In: *arXiv preprint arXiv:2203.05736*.

Lin, Chin-Yew (2004). "ROUGE: A Package for Automatic Evaluation of Summaries". In: *Text Summarization Branches Out*, pp. 74–81.

Lippmann, Richard P (1989). "Review of neural networks for speech recognition". In: *Neural computation* 1.1, pp. 1–38.

Lowerre, B and R Reddy (1976). "The harpy speech recognition system: performance with large vocabularies". In: *The Journal of the Acoustical Society of America* 60.S1, S10–S11.

Lu, Liang et al. (2020). "Exploring transformers for large-scale speech recognition". In: *arXiv preprint arXiv:2005.09684*.

Lu, Yiping et al. (2019). "Understanding and Improving Transformer From a Multi-Particle Dynamic system Point of View". In: *arXiv preprint arXiv:1906.02762*.

Maddison, Chris J, Daniel Tarlow, and Tom Minka (2014). "A$\star$ Sampling". In: *Advances in Neural Information Processing Systems* 27.

Maekaku, Takashi et al. (2021). "Speech representation learning combining conformer cpc with deep cluster for the zerospeech challenge 2021". In: *arXiv preprint arXiv:2107.05899*.

Martin, Thomas B, AL Nelson, and HJ Zadell (1964). *Speech Recognition by feature-Abstraction Techniques*. Tech. rep. Raytheon Co. Waltham Mass.

McCulloch, Warren S and Walter Pitts (1943). "A logical calculus of the ideas immanent in nervous activity". In: *The bulletin of mathematical biophysics* 5.4, pp. 115–133.

Meng, Yen et al. (2021). "Don't speak too fast: The impact of data bias on self-supervised speech models". In: *arXiv preprint arXiv:2110.07957*.

Meng, Zhong et al. (2021). "Internal Language Model Adaptation with Text-Only Data for End-to-End Speech Recognition". In: *arXiv preprint arXiv:2110.05354*.

Mermelstein, Paul (1976). "Distance measures for speech recognition, psychological and instrumental". In: *Pattern recognition and artificial intelligence* 116, pp. 374–388.

Nagata, K (1963). "Spoken digit recogniser for Japanese language". In: *NEC research & development* 6.

Narayanan, Arun et al. (2019). "Recognizing long-form speech using streaming end-to-end models". In: *arXiv preprint arXiv:1910.11455*.

Olson, Harry F and Herbert Belar (1956). "Phonetic typewriter". In: *The Journal of the Acoustical Society of America* 28.6, pp. 1072–1081.

Panayotov, Vassil et al. (2015). "Librispeech: an asr corpus based on public domain audio books". In: *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, pp. 5206–5210.

Papineni, Kishore et al. (2002). "BLEU: A Method for Automatic Evaluation of Machine Translation". In: *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pp. 311–318.

Pratap, Vineel, Awni Hannun, et al. (2018). "wav2letter++: The fastest open-source speech recognition system". In: *arXiv preprint arXiv:1812.07625*.

Pratap, Vineel, Qiantong Xu, et al. (2020). "Mls: A large-scale multilingual dataset for speech research". In: *arXiv preprint arXiv:2012.03411*.

Rabiner, L et al. (1979). "Speaker-independent recognition of isolated words using clustering techniques". In: *IEEE Transactions on Acoustics, Speech and Signal Processing* 27.4, pp. 336–349.

Reddy, D Raj et al. (1977). "Speech understanding systems: a summary of results of the five-year research effort". In:

Riviere, Morgane et al. (2020). "Unsupervised Pre-training Transfers Well Across Languages". In: *ICASSP 2020*. IEEE, pp. 7414–7418.

Roach, Peter et al. (1996). "BABEL: An Eastern European multi-language database". In: *Proceeding of Fourth International Conference on Spoken Language Processing. ICSLP'96*. Vol. 3. IEEE, pp. 1892–1893.

Rosenblatt, Frank (1958). "The perceptron: a probabilistic model for information storage and organization of the brain". In: *Psychological Review* 65.6, p. 386.

Rumelhart, David E, Geoffrey E Hinton, and Ronald J Williams (1986). "Learning representations by back-propagating errors". In: *Nature* 323.6088, pp. 533–536.

Saito, Kuniaki et al. (2019). "Semi-supervised Domain Adaptation via Minimax Entropy". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8050–8058.

Sakai, T and S Doshita (1961). "Phonetic typewriter". In: *The Journal of the Acoustical Society of America* 33.11, pp. 1664–1664.

Sakoe, Hiroaki (1971). "Dynamic-programming approach to continuous speech recognition". In: *1971 Proc. the International Congress of Acoustics, Budapest.*

Sakoe, Hiroaki and Seibi Chiba (1978). "Dynamic programming algorithm optimization for spoken word recognition". In: *IEEE transactions on acoustics, speech, and signal processing* 26.1, pp. 43–49.

Sawata, Ryosuke, Yosuke Kashiwagi, and Shusuke Takahashi (2021). "Improving Character Error Rate Is Not Equal to Having Clean Speech: Speech Enhancement for ASR Systems with Black-box Acoustic Models". In: *arXiv preprint arXiv:2110.05968.*

Schneider, Steffen et al. (2019). "wav2vec: Unsupervised pre-training for speech recognition". In: *arXiv preprint arXiv:1904.05862.*

Sidorov, Maxim et al. (2013). "Survey of Automated Speaker Identification Methods". In: *2013 9th International Conference on Intelligent Environments*. IEEE, pp. 236–239.

Sukhadia, Vrunda N and S Umesh (2022). "Domain Adaptation of low-resource Target-Domain models using well-trained ASR Conformer Models". In: *arXiv preprint arXiv:2202.09167.*

Suzuki, Joji (1961). "Recognition of Japanese vowels". In: *Journal of the Radio Research Laboratory* 8.37.

Synnaeve, Gabriel and Emmanuel Dupoux (2016). "A temporal coherence loss function for learning unsupervised acoustic embeddings". In: *Procedia Computer Science* 81, pp. 95–100.

Synnaeve, Gabriel, Qiantong Xu, et al. (2019). "End-to-end asr: from supervised to semi-supervised learning with modern architectures". In: *arXiv preprint arXiv:1911.08460.*

Thomas, Samuel et al. (2022). "Integrating Text Inputs For Training and Adapting RNN Transducer ASR Models". In: *arXiv preprint arXiv:2202.13155.*

Tian, Jinchuan et al. (2022). "Improving Mandarin End-to-End Speech Recognition with Word N-gram Language Model". In: *arXiv preprint arXiv:2201.01995*.

Tian, Jingguang, Xinhui Hu, and Xinkang Xu (2022). "Royalflush Speaker Diarization System for ICASSP 2022 Multi-channel Multi-party Meeting Transcription Challenge". In: *arXiv preprint arXiv:2202.04814*.

Trinh, Viet Anh and Sebastian Braun (2021). "Unsupervised Speech Enhancement with speech recognition embedding and disentanglement losses". In: *arXiv preprint arXiv:2111.08678*.

Trinh, Viet Anh, Hassan Salami Kavaki, and Michael I Mandel (2021). "ImportantAug: a data augmentation agent for speech". In: *arXiv preprint arXiv:2112.07156*.

Valk, Jörgen and Tanel Alumäe (2020). "VoxLingua107: a Dataset for Spoken Language Recognition". In: *arXiv preprint arXiv:2011.12998*.

Vaswani, Ashish et al. (2017). "Attention Is All You Need". In: *arXiv preprint arXiv:1706.03762*.

Velichko, VM and NG Zagoruyko (1970). "Automatic recognition of 200 words". In: *International Journal of Man-Machine Studies* 2.3, pp. 223–234.

Vintsyuk, Taras K (1968). "Speech discrimination by dynamic programming". In: *Cybernetiks* 4.1, pp. 52–57.

Viterbi, Andrew (1967). "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm". In: *IEEE transactions on Information Theory* 13.2, pp. 260–269.

Waibel, Alex et al. (1989). "Phoneme recognition using time-delay neural networks". In: *IEEE transactions on acoustics, speech, and signal processing* 37.3, pp. 328–339.

Wang, Changhan et al. (2021). "Voxpopuli: A large-scale multilingual speech corpus for representation learning, semi-supervised learning and interpretation". In: *arXiv preprint arXiv:2101.00390*.

Wang, Quan et al. (2022). "Attentive Temporal Pooling for Conformer-based Streaming Language Identification in Long-form Speech". In: *arXiv preprint arXiv:2202.12163*.

Wu, Zhanghao et al. (2020). "Lite Transformer with Long-Short Range Attention". In: *arXiv preprint arXiv:2004.11886*.

Xu, Liyan et al. (2022). "RescoreBERT: Discriminative Speech Recognition Rescoring with BERT". In: *arXiv preprint arXiv:2202.01094*.

Xu, Qiantong et al. (2021). "Self-training and pre-training are complementary for speech recognition". In: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 3030–3034.

Yadav, Hemant and Sunayana Sitaram (2022). "A Survey of Multilingual Models for Automatic Speech Recognition". In: *arXiv preprint arXiv:2202.12576*.

Yang, Chao-Han Huck et al. (2022). "Mitigating Closed-model Adversarial Examples with Bayesian Neural Modeling for Enhanced End-to-End Speech Recognition". In: *arXiv preprint arXiv:2202.08532*.

Yang, Shu-wen et al. (2021). "SUPERB: Speech Processing Universal Performance Benchmark". In: *arXiv preprint arXiv:2105.01051*.

Yang, Yao-Yuan et al. (2021). "Torchaudio: Building blocks for audio and speech processing". In: *arXiv preprint arXiv:2110.15018*.

Ye, Guoli et al. (2021). "Have best of both worlds: two-pass hybrid and E2E cascading framework for speech recognition". In: *arXiv preprint arXiv:2110.04891*.

Yeh, Ching-Feng et al. (2019). "Transformer-transducer: End-to-end speech recognition with self-attention". In: *arXiv preprint arXiv:1910.12977*.

Yu, Fan et al. (2021). "M2MeT: The ICASSP 2022 Multi-Channel Multi-Party Meeting Transcription Challenge". In: *arXiv preprint arXiv:2110.07393*.

Zeghidour, Neil and David Grangier (2021). "Wavesplit: End-to-end Speech Seapration by Speaker Clustering". In: *IEEE/ACM Transactions on Audio, Speech and Language Processing* 29, pp. 2840–2849.

Zeineldeen, Mohammad et al. (2021). "Conformer-based Hybrid ASR System for Switchboard Dataset". In: *arXiv preprint arXiv:2111.03442*.

Zhang, Zhengyi and Pan Zhou (2022). "End-to-end contextual asr based on posterior distribution adaptation for hybrid ctc/attention system". In: *arXiv preprint arXiv:2202.09003*.

Zhou, Wei et al. (2021). "On Language Model Integration for RNN Transducer based Speech Recognition". In: *arXiv preprint arXiv:2110.06841*.