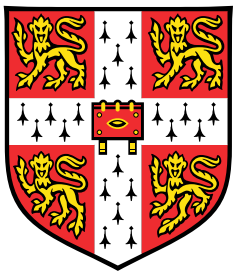
Distilling and Forgetting in Large Pre-Trained Models



Tony Wu

Supervisors: Prof. Mark Gales
Dr. Mengjie Qian
Adian Liusie

Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of
Master of Philosophy in Machine Learning and Machine Intelligence

Sidney Sussex College August 2023

I would like to dedicate this thesis to the people who made this journey meaningful.

Declaration

I, Tony Wu of Sidney Sussex College, being a candidate for the MPhil in Machine Learning and Machine Intelligence, hereby declare that this report and the work described in it are my own work, unaided except as may be specified below, and that the report does not contain material that has already been used to any substantial extent for a comparable purpose.

Word count: 14,263

Tony Wu
August 2023

Declaration

The core code - module excluded - is entirely my own. The vast majority of the implementation is carried out in the Python software language (version 3.10.11). Extensively used Python libraries include:

- Standard packages: `numpy` 1.24.3, `typer` 0.9.0, `jiwer` 3.0.2, `pandas` 2.0.2
- Plotting packages: `matplotlib` 3.7.1, `seaborn` 0.12.2
- Deep learning packages: `torch` 2.0.1, `transformers` 4.31, `datasets` 2.13.0, `accelerate` 0.20.3, `optimum` 3.2.0, huggingface-hub 0.15.1
- Experiment tracking package: `wandb` 0.15.4

In particular, the Whisper model was implemented by `transformers` and contributed by Arthur Zucker. The weights of the pre-trained vanilla Whisper models have been released by OpenAI.

The entire code developed for this thesis can be found in the following GitHub repository: https://github.com/tonywu71/distilling-and-forgetting-in-large-pre-trained-models/releases/tag/v1.0.0.

Experiments were run on Cambridge's High Performance Computing Service (HPC). Note that extra storage space was granted by ALTA (Cambridge University Institute for Automated Language Teaching and Assessment) for storing the datasets used in this project.

Tony Wu
August 2023

Acknowledgements

I want to thank Prof. Mark Gales for his guidance and mentorship throughout the writing of this dissertation. As I was never exposed to proper research before working on this thesis, his expertise in instilling research rigor has been invaluable to my academic growth. I was particularly impressed by his eloquence and communication clarity, which I tried to hone during these five months of research. I would also like to sincerely thank Dr. Mengjie Qian, Adian Liusie, and Rao Ma for providing me with invaluable direction and technical support during the five months spent on this thesis. I would also like to thank my friends and family for making this journey meaningful. Julien, Tam-Phuc, Stéphanie, Loan, Timothé, Daphné, Louis, Claire, Martin, Marcel, Matheus, João, Mazel, Alex, Marco, Henry, and Selina, I am indebted for your unconditional support. As for Jonathan, Andrey, Samira, and Phil, I am incredibly grateful for having the chance to grow under your mentorship.

Abstract

Large pre-trained models have become popular for deep learning applications because of their impressive performance. Moreover, they can be fine-tuned to perform specific tasks with relatively small amounts of data. However, many challenges remain to be tackled. Notably, the substantial number of parameters in these models makes them computationally expensive and slow at inference. Additionally, a separate issue arises when fine-tuning models to incorporate new tasks, as this process often leads to the forgetting of previously acquired knowledge. This is problematic as a robust and generic model is desirable. This dissertation investigates methods to solve these two issues. The application of these techniques is tested on Whisper, a state-of-the-art Automatic Speech Recognition (ASR) foundation model with a Transformer architecture.

Firstly, the trade-off between model size and performance can be solved by knowledge distillation. Distillation is a machine learning training technique used to transfer knowledge from a large model (the *teacher*) to a smaller one (the *student*). At its core, knowledge distillation involves training the student to mimic the teacher's output logits. This thesis explores two unsupervised distillation methods suited for sequence-to-sequence models: word-level distillation and K-best sequence-level distillation. The models are trained exclusively on AMI, a 100h-long English conversational dataset. Using Whisper `tiny` as the student and Whisper `medium` as the teacher, it is shown that naively using the raw teacher outputs for 1-best distillation gives a poor word error rate (WER) improvement from 27.74% to 27.30% for 1-best distillation on the AMI test set. Therefore, normalizing the teacher's predictions and filtering out hallucinations prior to training are investigated to improve the distillation performance. In particular, filtering out the teacher's transcriptions with high values of gzip compression ratio is demonstrated to be quite effective: while it removes only 0.08% of the examples and 0.26% of the audio from the training set, the subsequent 1-best yields a much more reasonable WER decrease from 27.74% to 22.80% on AMI test, i.e. a 17.81% relative improvement.

Secondly, continual learning can be achieved using Elastic Weight Consolidation (EWC). By estimating the Fisher information matrix for the previous tasks, it is possible to measure

the task-related importance of the model's parameters, which EWC can *a fortiori* adequately regularize during fine-tuning. Furthermore, Task Alignment Consolidation (TAC) - a novel method introduced in this thesis - considers regularization from the prediction space to preserve the multilingual capabilities of Whisper without needing data from the previous tasks. Nonetheless, this method is shown to be inefficient. On the other hand, EWC proves to be an impressive candidate for continual learning. Not only does it manage to keep the relative WER increase for French transcription under 4% of the vanilla performance for the whole training, but it also significantly reduces forgetting for other non-English transcription tasks: compared to default fine-tuning, EWC achieves an average relative WER drop of 25.13% on the non-English datasets from Multilingual LibriSpeech.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

Large pre-trained models are a subset of deep learning models trained on massive amounts of data. They also exhibit a remarkable capacity to solve complex problems like understanding human text or generating human-like images. At the time of writing, the trend is to go with larger and larger pre-trained models. This tendency is motivated by the scaling laws introduced by OpenAI in [35] and can be witnessed for speech as well (see Figure 1.1, [62, 43, 24, 5, 32, 14, 72, 13, 4, 57, 55]). However, many challenges remain to be tackled. Notably, the substantial number of parameters in these models makes them computationally expensive and slow at inference. Additionally, a separate issue arises when fine-tuning models to incorporate new tasks, as this process often leads to the forgetting of previously acquired knowledge. This is problematic as a robust and generic model is desirable. This dissertation investigates methods to solve both these issues.

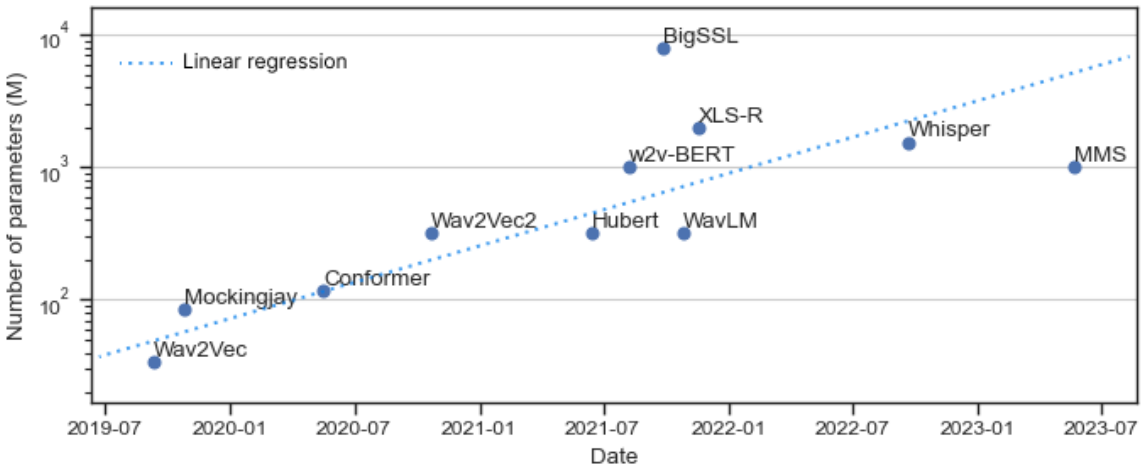


Fig. 1.1 Timeline of Transformer-based large pre-trained models for speech. For models with several sizes, the largest was considered.

The focus of this thesis is on Automatic Speech Recognition (ASR) large pre-trained models, which aim at transcribing spoken words into written text. In September 2023, Radford et al. [57] introduced Whisper, the state-of-the-art ASR model at the time of writing. For this reason, we chose to focus on Whisper for this thesis.

The first challenge of interest is the trade-off between performance and model size. As large pre-trained models have a substantial number of weights, they are computationally expensive, too slow for real-time applications, and too heavy to deploy on mobile devices with limited resources. Reducing the size of a model can be achieved with model compression techniques such as weight pruning [26, 25, 74, 61], low-rank weight decomposition [12], parameter sharing [41, 36], quantization [34], and binarization. Knowledge distillation, first introduced by Hinton, [29], takes a different approach: it aims to transfer the knowledge from a large model (the *teacher*) to a smaller one (the *student*). At its core, distillation teaches the student to mimic the output logits of the teacher for a classification task. This form of training is effective as the teacher's logits (*soft labels*) contain additional information about the model's uncertainty compared to fine-tuning, which only provides a binary reference (*hard labels*). As knowledge distillation has proved very successful in the literature [60, 45, 65], it will be the method of interest in this thesis. Two unsupervised distillation methods are explored: the word-level distillation and the K-best sequence-level distillation [38]. Word-level distillation consists of applying distillation for each element of the reference sequence. However, inference uses the tokens the model has generated at previous time steps as the previous ground-truth target tokens are no longer available. Consequently, this discrepancy between training and generation during scoring is likely to hamper the performance of the trained model [7]. For this reason, sequence-level distillation generates K sequences using beam search with the teacher model and then teaches the student to mimic the teacher's probabilities over these K sequences.

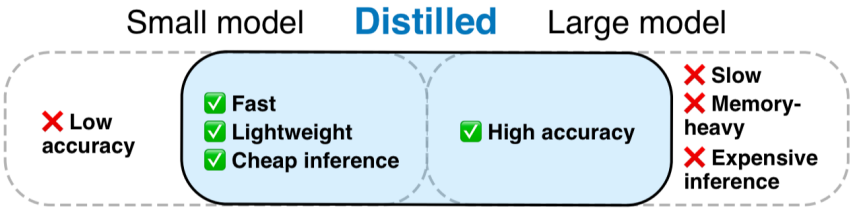


Fig. 1.2 Knowledge distillation solves the trade-off between performance and model size.

The second challenge is about achieving continual learning (also known as lifelong learning or incremental learning), which aims to train models to progressively acquire and

integrate new knowledge over time without forgetting previously learned information. While this is particularly interesting for adapting to new data trends and for preventing unnecessary training, continual learning is non-trivial since fine-tuning a pre-trained model for a new task usually causes catastrophic forgetting [48, 58, 19]. A few popular continual learning methods comprise *Learning without Forgetting* [42], *Memory Aware Synapses* [2], or to simply interleave examples from the old datasets within the new one [10]. This thesis will focus on Elastic Weight Consolidation (EWC) [39]. EWC uses the observed Fisher information matrix to identify important parameters relative to the previous tasks during the pre-training phase. Fine-tuning is then modified to prevent large deviations in these important parameters. Moreover, we have contributed to a new method called Task Alignment Consolidation (TAC). This novel method considers regularization from the prediction space without needing data used for pre-training. To be more precise, using the original copied model and the currently trained one, it transcribes audio spoken in language X while being prompted to transcribe using a different language Y, resulting in words in Y that sound similar to the reference text in X. Then, at each training step, the currently trained model is taught to mimic the pseudo-transcriptions of the original model.

Besides, this work could be used to develop a new language assessment tool ALTA (Cambridge University Institute for Automated Language Teaching and Assessment). By distilling Whisper on the ALTA's non-native English ASR dataset, we aim to create a cost-efficient model for speech recognition while maintaining high-performance standards. Eventually, preserving the multilingual capabilities of Whisper should prove interesting for transcribing speech from people alternating between different languages (*code-switching*).

This report comprises six other chapters: Chapter 2 describes ASR in greater detail and looks at the technical details of Whisper. Chapter 3 defines knowledge distillation and introduces the different methods that will be assessed throughout this thesis. Chapter 4 introduces the concept of continual learning and highlights two methods to prevent forgetting during fine-tuning. Chapter 5 discusses the experimental setup created before the experiments, including the datasets used and how these will be evaluated. Chapter 6 is the results and discussion section, which displays the overall results for unsupervised distillation and continual learning. Finally, Chapter 7 is the conclusion section which rounds off the main findings of this thesis and highlights promising lines for future work.

Chapter 2

Automatic Speech Recognition

2.1 Background

2.1.1 Feature extractor

Speech - like all sounds - is a wave that propagates through a medium e.g. the air. To be processed, stored, and used by a computer, speech must be converted into discrete values. A common and effective discrete representation for speech is the mel spectrogram, adapted to the human ear's non-linear frequency response. To create a mel spectrogram, the audio is split into short audio frames. Then, a Fourier Transform algorithm is used to obtain a spectrogram. Finally, the spectrogram frequencies are mapped to the mel scale, which is performed by multiplying the spectrogram by a set of filters called mel filterbanks at each time step. As the example from Figure 2.1 shows, the spectrogram signal is more uniformly spread at each time step.

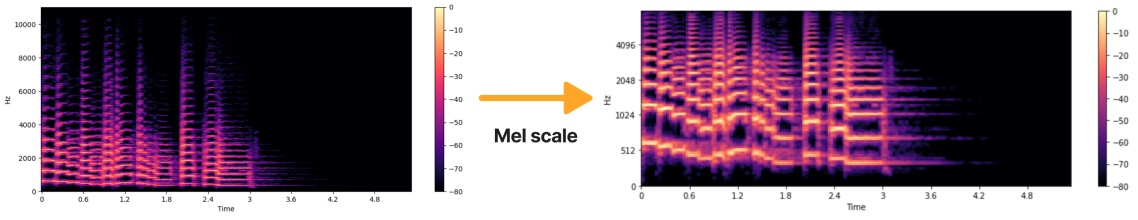


Fig. 2.1 Illustration of the mel scale transformation.

2.1.2 Tokenizer

Because machine learning models can only process numbers, we need to transform text inputs into numerical data. Although character-to-index mapping can achieve this, one can hypothesize that a character on its own contains almost no signal about the semantics of a sentence. Thus, we are interested in finding the most meaningful text representation for a given model and, if possible, the smallest representation. Tokenization - the process of breaking down a text into smaller units - provides such a solution. These units are called tokens and can be words, characters, or subwords, depending on the type of tokenization strategy. In particular, the BPE tokenizer [64] is trained by iteratively merging the most frequent pairs of subwords - initially all Unicode characters present in the training set - until the desired vocabulary size is reached.

2.1.3 End-to-end ASR models

End-to-end models are popular for ASR as they directly convert spoken language into text without intermediate steps, making the pipeline simpler and quite often more performant. Three popular approaches comprise the Connectionist Temporal Classification (CTC) models [23], the Recurrent Neural Network Transducer (RNN-T) [22], and attention-based models [6, 67].

CTC

CTC models like Wav2Vec [62], Wav2Vec2 [5], HuBERT [32], and WavLM [13] achieved competitive performances over the last few years. At its core, a CTC model operates on a one-to-one mapping between audio frames and outputs and achieves alignment by repeating symbols (often phonemes) and inserting blank symbols. However, CTC wrongly assumes independence between the different outputs and an external language model has to be manually added through interpolation or shallow fusion for better performance.

RNN-T

RNN-T solves the independence issue of the CTC models by introducing an encoder-predictor-joiner architecture. While the encoder generated features for the current audio frame, the predictor is in charge of producing features from the previously generated outputs. The joiner then combines the encoder's and the predictor's features to either predict a new element for the generated sequence while keeping the focus on the current audio frame or to skip output generation and focus on the next audio frame at the next time step. Note

that ontrarily to CTC and attention-based models, RNN-T is suitable for online inference [27, 73].

Attention-based

Attention models have an encoder-decoder architecture designed to handle sequential data to effectively capture long-range dependencies and patterns thanks to self-attention. As the name suggests, self-attention computes attention weights for the input sequence elements with respect to themselves which is not possible for the RNN-T.

2.2 Whisper

Whisper, an open-source end-to-end ASR model developed by OpenAI [57] and released in September 2023, will be the model of interest in this thesis. It was pre-trained on a massive 680,000-hour dataset of diverse audio and is also a multitasking model that can perform multilingual speech recognition, speech translation, language identification, voice activity detection, and utterance timestamping.

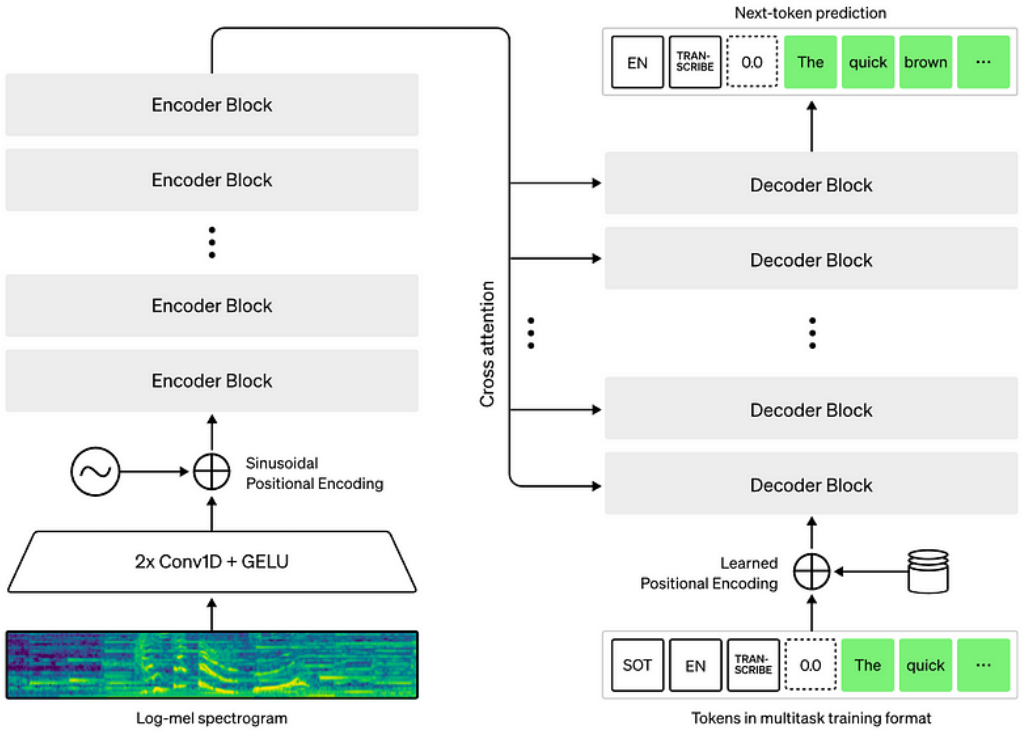


Fig. 2.2 Whisper's architecture as presented by Radford et al. [57].

2.2.1 Architecture

Whisper is based on the Transformer's architecture. Thus, it takes as input an audio spectrogram obtained using a mel spectrogram and outputs a sequence of text tokens obtained using a BPE tokenizer. More precisely, the encoder encodes the spectrogram to form a sequence of hidden states. Finally, the decoder predicts text tokens in an auto-regressive fashion while being conditioned on previously generated tokens and the encoder's hidden states. This process is shown in Figure 2.2). Finally, it is interesting to highlight that Whisper's encoder is equivalent to an acoustic model and the decoder to an implicit language model because it processes tokens in an auto-regressive way.

2.2.2 Multi-task pre-training

Whisper's vocabulary comprises several special tokens that are used to implement multi-tasking during training and inference. First, the start-of-transcript token is followed by a language identification and a task special token, where the task can be either about transcription or translation from a supported language to English. Second, OpenAI added a specific format to tell Whisper to predict timestamps (triggered by removing the `<|notimestamps|>` token) and to detect voice activity (`<|nospeech|>`).

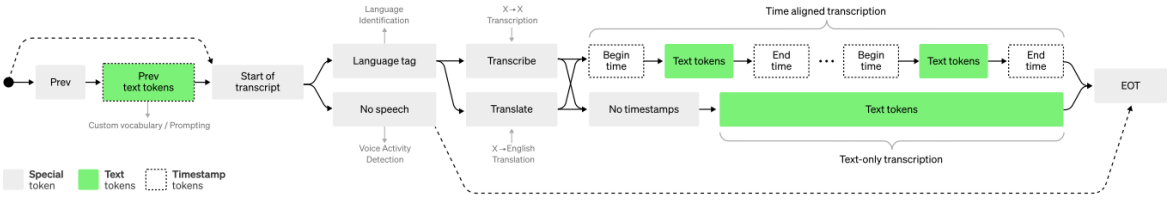


Fig. 2.3 Whisper's multi-task format as presented by Radford et al. [57].

The Whisper model itself was trained using the regular teacher-forced cross-entropy. As a reminder, teacher forcing is a technique where the ground-truth output from the previous time step is used as input to the decoder at the current time step. This guarantees that the model probabilities and the ground-truth labels are compared using the same history.

2.2.3 Trade-off between model size and latency

OpenAI released several pre-trained Whisper models with different sizes, ranging from 39M parameters for the smallest model (`tiny`) to 1.55B for the largest (`large-v2`). As we are interested in having a model with quick inference time, we decided to benchmark the mean inference time of the different Whisper models. The methodology was the following. First,

we picked a fixed arbitrary audio sample. Then, for each model size, we first ran inference with greedy decoding 10 times to warm up the system and ensure that the model was properly loaded in the GPU. Then, we timed and ran the inference 100 times. The experiment was run on an Ampere A100 Nvidia GPU. Results are shown in Table 2.1.

Model	Layers	Width	Heads	Parameters (M)	Mean inference time (ms)
`tiny`	4	384	6	39	102.12 ± 12.47
`base`	6	512	8	74	133.90 ± 13.07
`small`	12	768	12	244	241.57 ± 0.72
`medium`	24	1024	16	769	452.83 ± 1.42
`large-v2`	32	1280	20	1550	661.02 ± 1.22

Table 2.1 Evolution of the number of the mean inference time with respect to the number of parameters in the different Whisper models.

2.2.4 Decoding strategies

Because of its auto-regressive nature, Whisper models the likelihood of a sequence of tokens $y_{1:T}$ with the following conditional probability:

$$p\left(y_{1:T} \mid \mathbf{Y_0}, \mathbf{x}_{1:L}\right) = \prod_{t=1}^{T} p\left(y_t \mid y_{1:t-1}, \mathbf{Y_0}, \mathbf{x}_{1:L}\right) \tag{2.1}$$

where $\mathbf{Y_0}$ is the initial prompt, and $\mathbf{x}_{1:L}$ is the audio feature extracted by the encoder. For Whisper, an example of an initial prompt is $\mathbf{Y_0}$ = (`<|startoftranscript|>`, `<|en|>`, `<|transcribe|>`) where the last two tokens indicate that we are interesting in transcribing English.

Decoding the optimal sequence for such a model is non-trivial for a few key reasons:

- For a vocabulary of size N and maximum sequence length T, there are $\mathcal{O}(N^T)$ possible sequences. Thus, exhaustively searching the whole space is intractable.
- The optimal choice for the next word requires looking several tokens ahead to see what words yield the best overall sequence.
- Choosing an incorrect word early on skews all future predictions down the wrong path. Thus, the decoding algorithm needs to be robust to its own mistakes.
- Decoding needs to happen quickly at inference time to be useful and cheap.

Keeping these reasons in mind, we will compare 3 methods of decoding for Whisper: greedy search, k-beam search, and sampling.

Greedy search

At inference time, greedy search selects the highest probability word predicted by the model each time it generates the next word. For instance, assume we have already generated a sentence $y_{1:t}$. The word y_{t+1} is picked such that:

$$\forall t > 1, y_{t+1} = \text{argmax}_y \, p\left(y \mid y_{1:t}\right) \tag{2.2}$$

Greedy search is fast as it runs in $\mathcal{O}(NT)$ where N is the vocabulary size and T is the sequence length. However, it only considers the single most likely word at each step rather than looking ahead to find the best overall sequence. This often leads to a sub-optimal sequence.

K-beam search

Instead of selecting the highest probability token at each decoding step like in greedy search, beam search decoding maintains a list of the K most probable next tokens, where K is called the beam size. The next set of beam sequences is chosen by considering all possible next-token extensions of the existing set and by selecting the K extensions with the highest probabilities. The process is repeated until we reach the maximum length or an end-of-sequence (EOS) token, and the most likely sequence is returned.

Sampling

A simple method to generate more human-like sentences is to randomly sample the next token y_t of a sequence according to the probability distribution p of the model. This can be expressed by Equation 2.3 using the same notations as Equation 2.1.

$$\forall t \geq 1, y_{t+1} \sim p\left(y \mid y_{1:t}, \mathbf{Y_0}, \mathbf{x}_{1:L}\right) \tag{2.3}$$

Sampling is generally used in combination with temperature [31], top-k sampling [18] and top-p sampling [31] to prevent rare and incoherent tokens from being sampled.

2.2.5 Known issues

The pre-trained Whisper models tend to generate punctuation, capitalization, and phrases with inverse text normalization (e.g. it would format addresses, time, dates, numbers, and abbreviations). Moreover, they often remove disfluencies (e.g. repetitions, stutters, hesitations...). While this choice of formatting improves the readability of the output transcriptions,

there is a risk of a formatting misalignment with a given dataset. Ma et al. [46] have also demonstrated that the previous issues can be easily solved by fine-tuning or soft-prompting (training a few embeddings that got prepended to the decoder inputs) with limited available data.

As a generative model, Whisper can sometimes generate non-sense transcriptions referred to as hallucinations. In particular, one common form of hallucination consists of repetitions [31]. We hypothesize that as rare as they might be, hallucinations can severely hamper distillation as the student could be taught to hallucinate further. For reference, examples of such issues are shown in Table 2.2. Finally, it is worth mentioning that repetition-based hallucinations don't affect CTC models because of their one-to-one alignment between audio frames and outputs.

Category	**Reference**	**Whisper output**
Casing/punctuation	no no no no let's do the lounge corner	No, no, no, no. Let's do the lounge corner.
Inverse text normalization	twenty percent of fifteen dollars seventy three	20% of $15.73
Disfluencies	uh um it's uh easier to uh to lose uh th things uh like that on the computer uh	it's easier to do things like that on the computer
Hallucinations	yeah that's quite impressive actually	yes yes yes yes yes yes yes ... yes

Table 2.2 Examples of issues with Whisper's transcriptions.

Chapter 3

Knowledge distillation

At the time of writing, the trend in deep learning is to go with larger and larger pre-trained models. This assumption holds true for ASR models as well (see Figure 1.1 from chapter 1). In order to make inference less expensive, we are interested in model compression methods. Because recent studies [50, 69] have proved that parameters were usually redundant in pre-trained models, the first idea that comes to mind is to remove the unnecessary weights of a model by setting them to zero. This process is called pruning and has been widely explored in the literature [26, 25, 74, 61]. Additionally, low-rank weight decomposition [12] and parameter sharing [41, 36] can reduce the total number of parameters of the model. Quantization [34] approaches the problem from a different angle. While the number of weights is untouched, quantization proposes to represent model parameters with low-precision 8-bit integers instead of the usual 32-bit floating points. Not only do quantized weights take less space, but inference can be performed entirely with faster integer arithmetic. Model binarization, a more extreme version of quantization as weights are turned into booleans, was also demonstrated to be successful in the literature [15, 45]. Knowledge distillation, first introduced by Hinton, [29], takes a different approach: it aims to transfer the knowledge from a large model (the *teacher*) to a smaller and more weight-efficient one (the *student*). For a classification task, the student is trained to mimic the teacher's class distribution. As the teacher's distribution (*soft labels*) contains additional information about the relationships between classes and the model's uncertainty, the distillation signal is much richer than during fine-tuning with cross-entropy which is based on a binary reference (*hard labels*). An example of successful knowledge distillation is DistilBERT, a distilled version of BERT with only 60% of its size but 97% of its language understanding performance [60]. This thesis will focus on knowledge distillation, as we want to take advantage of the different Whisper sizes pre-trained by OpenAI. Note that, to the best of our knowledge and at the time of writing, only Shao et al. have published about distillation on Whisper [65].

3.1 Background

Knowledge distillation was introduced by Hinton for classification tasks [29]. This method suggests training the student (distribution p) to mimic the teacher model (distribution q). Compared to the default cross-entropy loss, which employs one-hot distribution at the reference class (*hard* probabilities), distillation is interesting as it involves using the teacher's distribution (*soft* probabilities) over all classes. This provides more insights about other classes associated with a given data point, e.g. two classes with similar scores are more likely to share common characteristics.

Let $\mathbf{x} \in \mathbb{R}^d$ a vector of d-dimensional features, $\mathbf{z}(\mathbf{x}) = [z_1(\mathbf{x}) \dots z_K(\mathbf{x})] \in \mathbb{R}^K$ the logit output of the last layer of the teacher model with K the number of classes. To convert these logit scores into valid probabilities, let's introduce the softmax function $\phi : z_k(\mathbf{x}) \mapsto \frac{\exp(z_k(\mathbf{x}))}{\sum_j \exp(z_j(\mathbf{x}))}$. However, a well-trained teacher model is likely to generate highly confident output scores. Consequently, the student model may encounter difficulties in capturing the signal from the non-dominant class. To address this challenge, a temperature hyperparameter τ is introduced, which is applied to the teacher distribution q right before the softmax function. The bigger τ, the smoother the distribution (see Figure 3.1).

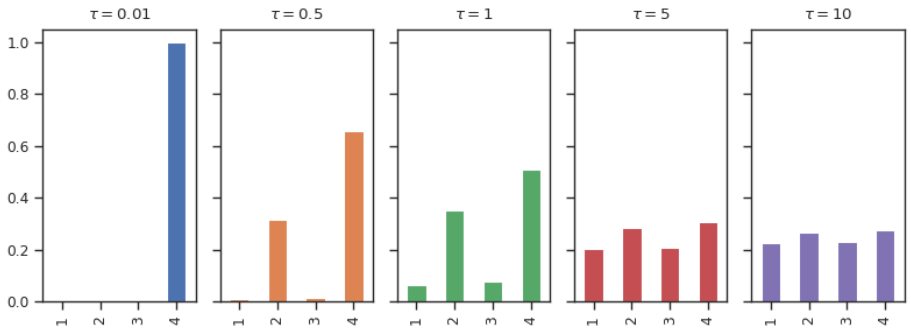


Fig. 3.1 Impact of the temperature τ on the distribution of the teacher's predictions $q'(y|\mathbf{x})$ for 4 classes. The original distribution is given for $\tau = 1$.

Hence, for a given example $\mathbf{x}$, we have for each class $k \in [|1, K|]$:

$$q'(y = k|\mathbf{x}) = \phi(z_k(\mathbf{x})) = \frac{\exp\left(z_k(\mathbf{x})/\tau\right)}{\sum_j \exp\left(z_j(\mathbf{x})/\tau\right)} \tag{3.1}$$

A few limiting behaviors are worth noticing:

- $\tau \sim 0^+$ yields the mode distribution $\underset{\mathbf{k} \in [|1,K|]}{\mathbb{1}} \left\{ k = \underset{\mathbf{j} \in [|1,K|]}{\text{argmax}} \left[q'_j(\mathbf{x}) \right] \right\}$

- $\tau = 1$ yields the original q distribution
- $\tau \to +\infty$ yields the uniform distribution $\mathscr{U}\{1, K\}$.

We now want to compare the two distributions q' and p. The Kullback-Leibler divergence $\mathscr{D}_{\mathrm{KL}}$ measures how different the 2 distributions are given a known example with features $\mathbf{x}$. Therefore, it is a good candidate for our loss:

$$\mathscr{D}_{\mathrm{KL}}(q'(\cdot|\mathbf{x})\|p(\cdot|\mathbf{x})) = \sum_{k=1}^{K} q'(y=k|\mathbf{x}) \log\left(\frac{q'(y=k|\mathbf{x})}{p(y=k|\mathbf{x})}\right) \tag{3.2}$$

Because we will run back-propagation and gradient descent to train our model, a factor $1/\tau^2$ will appear in the gradient computation with respect to $\mathbf{x}$. Simply multiplying $\mathscr{D}_{\mathrm{KL}}(q'(\cdot|\mathbf{x})\|p(\cdot|\mathbf{x}))$ by τ^2 will compensate for the additional factor:

$$\mathscr{L}_{\mathrm{KD}} = \tau^2 \mathscr{D}_{\mathrm{KL}}(q'(\cdot|\mathbf{x})\|p(\cdot|\mathbf{x})) \tag{3.3}$$

Empirically, it was shown that adding a weighted cross-entropy loss term $\mathscr{L}_{\mathrm{CE}}$ function of the training set gives better results as the teacher can still make wrong predictions:

$$\mathscr{L}_{\mathrm{student}} = \alpha \mathscr{L}_{\mathrm{CE}} + (1-\alpha)\mathscr{L}_{\mathrm{KD}} \tag{3.4}$$

The entire process is summarized in Figure 3.2.

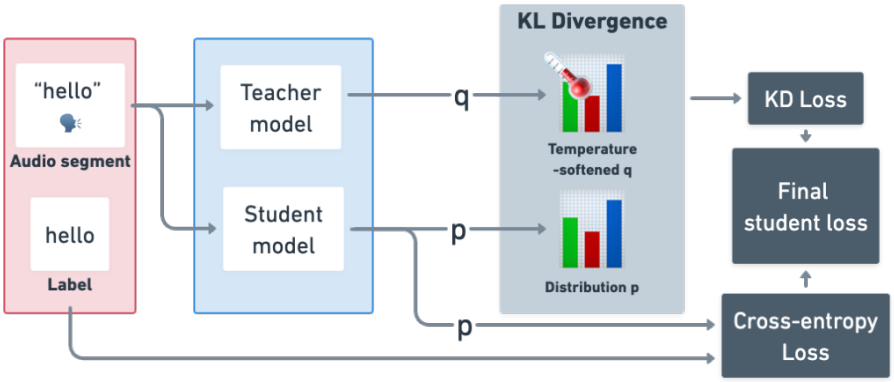


Fig. 3.2 Knowledge distillation process for a supervised classification task.

3.2 Word-level distillation

Word-level knowledge distillation is a straightforward first approach to adapting Hinton's distillation to a sequence-to-sequence model. It consists of using matching the token distribution at each time step in a teacher-forced fashion. The process is mathematically described in Equation 3.5 and can be visualized in Figure 3.3.

$$\mathscr{L}_{\text{WORD-KD}} = \tau^2 \sum_{t=1}^{T} \sum_{k=1}^{|\mathscr{V}|} \left[p'\left(y_t = k \mid \mathbf{x}_{1:L}, y_{<t}\right) \log \left(\frac{p'\left(y_t = k \mid \mathbf{x}_{1:L}, y_{<t}\right)}{q\left(y_t = k \mid \mathbf{x}_{1:L}, y_{<t}\right)} \right) \right] \tag{3.5}$$

with:

- $\mathbf{x}_{1:L}$ the sequence of audio features
- T the target length
- $\mathscr{V}$ the target vocabulary set
- $y = [y_1, \ldots, y_T]$ be random variable sequences representing the target sentence
- $\mathscr{S}$ the set of all possible sequences
- τ is the temperature
- q' the teacher distribution softened with temperature τ
- p the student distribution

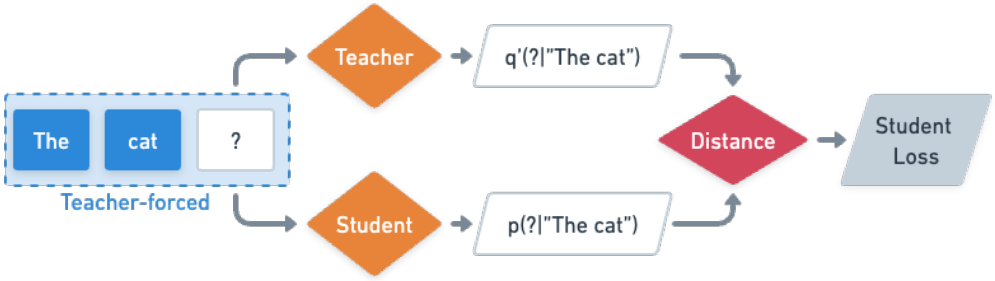


Fig. 3.3 The word-level distillation process (at each time step).

Interpolating this new loss term with cross-entropy loss, we get:

$$\mathscr{L}_{\text{student}} = \alpha \mathscr{L}_{\text{CE}} + (1 - \alpha) \mathscr{L}_{\text{WORD-KD}} \tag{3.6}$$

While word-level distillation is normally used for supervised training, we can also easily adapt them to work in an unsupervised fashion. Doing so is straightforward as we simply need to replace the references used for the cross-entropy loss with the teacher's predictions.

3.3 Sequence-level distillation

The problem with word-level distillation is that it fails to take the sequential aspect of the ASR outputs into account. In particular, the disparity between training (teacher-forced) and inference (using the previously generated token) can lead to errors that accumulate rapidly throughout the generated sequence [7]. To tackle this issue, we will consider sequence-level knowledge distillation introduced by Kim et al. [38] with a new loss function $\mathscr{L}_{\text{SEQ-KD}}$ that involves the probability over all possible sentences. This loss function is defined in Equation 3.7. The same mathematical notations from section 3.2 will be reused here. $\mathbf{Y}$ will be used instead of $y_{1:T}$ as the length of the output sequence isn't fixed here.

$$\mathscr{L}_{\text{SEQ-KD}} = -\sum_{\mathbf{Y}\in\mathscr{S}} [q(\mathbf{Y} \mid \mathbf{x}_{1:L}) \log p(\mathbf{Y} \mid \mathbf{x}_{1:L})] \tag{3.7}$$

Interpolating this new loss term with cross-entropy loss, we get:

$$\mathscr{L}_{\text{student}} = \alpha\mathscr{L}_{\text{CE}} + (1-\alpha)\mathscr{L}_{\text{SEQ-KD}} \tag{3.8}$$

Note that Kim et al. only considered sequence-level KD with $\alpha = 0$ as they raised concerns about the reference and the teacher predictions to be samples from quite different distributions. Consequently, we will only consider $\alpha = 0$ in our experiments, except if we state otherwise. The process can be visualized for each generated sentence in Figure 3.4.

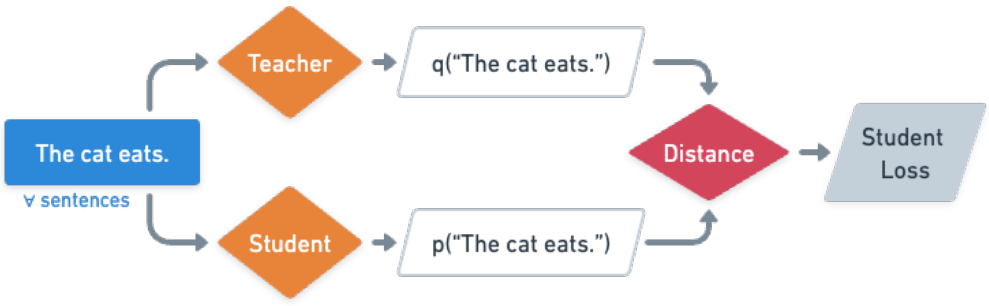


Fig. 3.4 The sequence-level distillation process.

Notice that the sum for the $\mathscr{L}_{\text{WORD-KD}}$ in Equation 3.5 contains an infinite number of terms as a sequence can be arbitrarily long. Thus, the sequence-level loss is intractable.

3.3.1 1-best distillation

To solve the intractability problem, we can approximate the teacher distribution q with its mode [38].

$$q(\mathbf{Y} \mid \mathbf{x}_{1:L}) \sim \mathbb{1}\left\{\mathbf{Y} = \underset{\mathbf{Y}' \in \mathscr{S}}{\operatorname{argmax}} \left[q\left(\mathbf{Y}' \mid \mathbf{x}_{1:L}\right)\right]\right\} \tag{3.9}$$

Let $\hat{\mathbf{Y}}$ be the output of a N-beam search on $q(\cdot|\mathbf{x}_{1:L})$. Using the new q approximation in Equation 3.7 holds:

$$\begin{aligned} \mathscr{L}_{\text{SEQ-KD}} &\approx -\sum_{\mathbf{Y} \in \mathscr{S}} \left[\mathbb{1}\{\mathbf{Y} = \hat{\mathbf{Y}}\} \log p(\mathbf{Y} \mid \mathbf{x}_{1:L})\right] \\ &\approx -\log p(\mathbf{Y} = \hat{\mathbf{Y}} \mid \mathbf{x}_{1:L}) \end{aligned} \tag{3.10}$$

This approximation yields what will refer to as the 1-best sequence-level distillation. In particular, the distillation loss is no longer intractable as the only expensive needed operation is to approximate the mode using N-beam search. Note that 1-best KD is equivalent to replacing the references with the teacher output sequences followed by training using the usual teacher-forced cross-entropy loss. Therefore, one might see 1-best KD as a form of unsupervised fine-tuning.

3.3.2 K-best distillation

To have a more precise approximation, we can also use the K-best sequences from an N-beam search to approximate the teacher distribution q, where $K \leq N$.

$$q(\mathbf{Y} \mid \mathbf{x}_{1:L}) \approx \begin{cases} \frac{q(\mathbf{Y}|\mathbf{x}_{1:L})}{\sum_{\mathbf{Y} \in \mathscr{S}_K} q(\mathbf{Y}'|\mathbf{x}_{1:L})} & \text{if } \mathbf{Y} \in \mathscr{S}_K \\ 0 & \text{otherwise} \end{cases} \tag{3.11}$$

where $\mathscr{S}_K$ is the N-best list from beam search.

Thus, we can rewrite $\mathscr{L}_{\text{SEQ-KD}}$ as:

$$\mathscr{L}_{\text{SEQ-KD}} \approx -\sum_{\mathbf{Y} \in \mathscr{S}_K} \left[\frac{q(\mathbf{Y} \mid \mathbf{x}_{1:L})}{\sum_{\mathbf{Y} \in \mathscr{S}_K} q\left(\mathbf{Y}' \mid \mathbf{x}_{1:L}\right)} \log p(\mathbf{Y} \mid \mathbf{x}_{1:L})\right] \tag{3.12}$$

This particular form of K-best knowledge distillation will be referred to as *uniform* K-best. A novel method introduced in this work is using rank-based exponential decay weighting for the K-best sequences. We will thus refer to this new method as *ranked* K-best.

$$\mathscr{L}_{\text{SEQ-KD}} \approx -\sum_{\mathbf{Y}\in\mathscr{S}_K}\left[w_k \frac{q(\mathbf{Y}\mid \mathbf{x}_{1:L})}{\sum_{\mathbf{Y}\in\mathscr{S}_K} q(\mathbf{Y}'\mid \mathbf{x}_{1:L})}\log p(\mathbf{Y}\mid \mathbf{x}_{1:L})\right] \tag{3.13}$$

where the w_k are defined by:

$$\forall k \in [|1,K|],\ w_k = \frac{e^{-\beta(k-1)}}{\sum_{l=1}^{K} e^{-\beta(l-1)}} \tag{3.14}$$

and where $\beta \in \mathbb{R}_+$ is a hyperparameter to tune. Note that the w_k are built so much that all the weights have values between 0 and 1, and their sum equals 1. The process for the K-best ranked KD is shown in Figure 3.5.

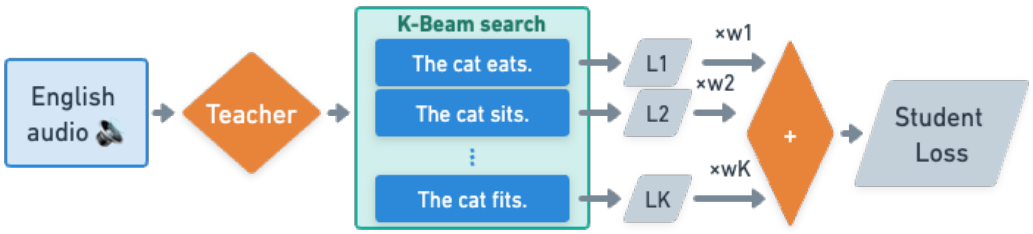


Fig. 3.5 The K-best distillation process.

Looking at Figure 3.6, the $\{w_k\}_{k\in[|1,K|]}$ gets closer to a discrete uniform distribution on $\mathscr{U}\{1,K\}$ when $\beta \to 0^+$. The equality holds when $\beta = 0$. When $\beta = 0$, the K-best uniform KD is - ignoring a multiplying factor - a special case of the K-best ranked KD. Finally, if using the same beam size N, K-best ranked KD is equivalent to 1-best KD when $\beta \to +\infty$.

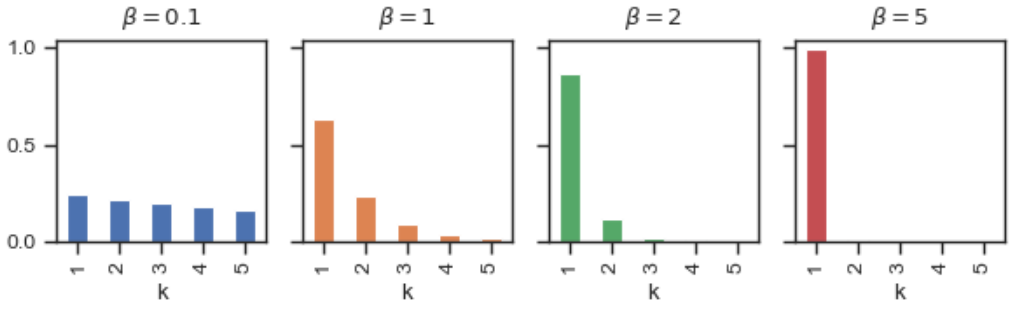


Fig. 3.6 Distribution of the weights w_k for the 5-best ranked approximation sequence-level distillation method and for different values of β.

Chapter 4

Continual learning

While knowledge distillation is a useful training method, the student model only learns one task and, just like fine-tuning, is susceptible to forgetting about what it had learned previously. This phenomenon is often referred to as catastrophic forgetting [19] and can be explained by the model adjusting its weights to minimize the loss for the new task, often leading to interference with the weights that encode information from previous tasks. Thus, we are interested in continual learning (also known as lifelong learning or incremental learning), which aims at training models to progressively acquire and integrate new knowledge over time without forgetting previously learned information.

In order to implement continual learning, it is possible to keep all or part of the previous datasets the model was trained on and to interleave examples within the new dataset [10]. For this reason, one can identify important parameters relative to the previous tasks during the pre-training phase. Fine-tuning would then be modified to prevent these important parameters from deviating too much. Elastic Weight Consolidation [39] implements this strategy by measuring the task importance of the weights using the observed Fisher information matrix. Another method is Memory Aware Synapses [2], which assesses the parameter's importance by measuring the impact of a small weight perturbation on the output. However, previous data is not always available, especially for large pre-trained models for which only the model weights get publicly released. Therefore, Learning without Forgetting [42] proposes to make a copy of the original model and to use it to force the newly trained one to match its predictions on the previous tasks. This thesis will first focus on Elastic Weight Consolidation (EWC).

4.1 Elastic Weight Consolidation

Elastic Weight Consolidation (EWC) is a technique introduced by Kirkpatrick et al. [39] to address catastrophic forgetting. At its core, EWC identifies the important parameters relative to the previous tasks learned during the pre-training phase. Using a regularization term weighted by the per-parameter importance, fine-tuning would then be modified to prevent these important parameters from deviating too much while allowing the network to adapt to new tasks using the less important weights. This is to contrast with other regularization methods like the L2 regularization which prevent overfitting, but often at the cost of performance for both the old and the new tasks. This process is illustrated in Figure 4.1.

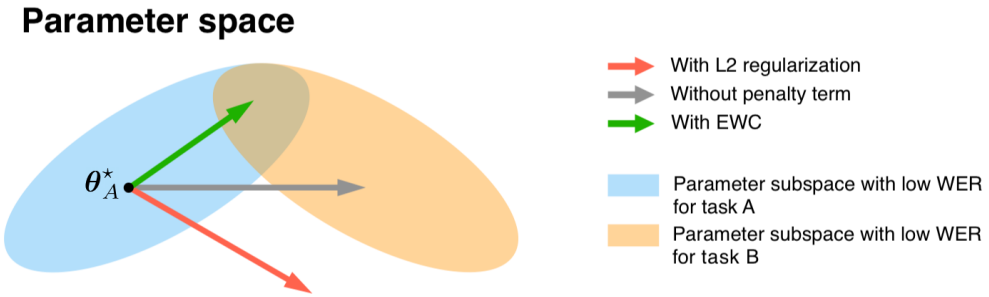


Fig. 4.1 Comparison of training trajectories using EWC. The intersection of the blue and orange subsets is where we want our model to have its parameters. $\theta_A^\star$ is the vector that contains the optimal weights for the model under task A.

4.1.1 Proof

To find out the form of a regularization term suitable for EWC, let's reformulate our original problem. We will consider two tasks to simplify our study: A is our original task and B is the new task we would like to train our model on. We will use $\mathscr{D}_A = \{\mathbf{x}_A^{(i)}, y_A^{(i)}\}_{i\in[|1,N_A|]}$ and $\mathscr{D}_B = \{\mathbf{x}_B^{(i)}, y_B^{(i)}\}_{i\in[|1,N_B|]}$ to refer to the data associated with the given task. Our goal is to maximize the posterior likelihood of the model weights $\theta \in \mathbb{R}^P$ knowing the data for both tasks A and B [1, 33]. The objective function is, therefore:

$$\max_{\theta\in\mathbb{R}^P} \{\ell(\theta) = \log(p(\theta \mid \mathscr{D}_A,\ \mathscr{D}_B))\} \tag{4.1}$$

Because A is our first task, we want to use the Bayes rule with respect to $p(\cdot|\mathscr{D}_A)$:

$$\begin{aligned}\log(p(\theta \mid \mathscr{D}_A,\, \mathscr{D}_B)) &= \log(p(\mathscr{D}_B \mid \mathscr{D}_A, \theta)) + \log(p(\theta \mid \mathscr{D}_A)) - \log(p(\mathscr{D}_B \mid \mathscr{D}_A)) \\ &= \log(p(\mathscr{D}_B \mid \theta)) + \log(p(\theta \mid \mathscr{D}_A)) - \log(p(\mathscr{D}_B \mid \mathscr{D}_A))\end{aligned} \tag{4.2}$$

In Equation 4.2, $\log(p(\mathscr{D}_B \mid \theta))$ can be obtained from the model's output logits and $\log(p(\mathscr{D}_B))$ is independent from θ. However, $\log(p(\theta \mid \mathscr{D}_A))$ has no explicit expression and is therefore intractable. Kirkpatrick et al. [39] propose to approximate the posterior likelihood for task A using a Laplace approximation based on the work of Mackay [47]. Let's derive this approximation to clearly understand the underlying assumptions of the Laplace approximation.

Let's denote the log-likelihood $\ell(\theta) = \log(p(\theta \mid \mathscr{D}_A))$ and $\theta_A^\star$ the weights of the model optimized under task A. The second-order Taylor expansion of $\ell(\theta)$ around $\theta_A^\star$ writes down as:

$$\begin{aligned}\ell(\theta) = \ell(\theta_A^\star) + \left(\frac{\partial \ell(\theta)}{\partial \theta}\bigg|_{\theta_A^\star}\right)(\theta - \theta_A^\star)^\top + \frac{1}{2}(\theta - \theta_A^\star)^\top \left(\frac{\partial^2 \ell(\theta)}{\partial^2 \theta}\bigg|_{\theta_A^\star}\right)(\theta - \theta_A^\star) \\ + \underset{\theta \sim \theta_A^\star}{\mathscr{O}}\left((\theta - \theta_A^\star)^2\right)\end{aligned} \tag{4.3}$$

Neglecting the terms of order 3 and more, and noting that $\frac{\partial \ell(\theta)}{\partial \theta}\Big|_{\theta_A^\star} = \mathbf{0}$, we get:

$$\ell(\theta) \approx \ell(\theta_A^\star) + \frac{1}{2}(\theta - \theta_A^\star)^\top \left(\frac{\partial^2 \ell(\theta)}{\partial^2 \theta}\bigg|_{\theta_A^\star}\right)(\theta - \theta_A^\star) \tag{4.4}$$

Going back to the non-log space, we can identify our probability density function with the density of a Gaussian-distributed random variable, such that:

$$p(\theta \mid \mathscr{D}_A) \sim \mathscr{N}(\theta_A^\star,\, \Sigma) \tag{4.5}$$

where:

$$\Sigma = \left(-\frac{\partial^2(\log(p(\theta \mid \mathscr{D}_A)))}{\partial^2 \theta}\bigg|_{\theta_A^\star}\right)^{-1} \tag{4.6}$$

To summarize, we approximated the intractable distribution $\log(p(\theta \mid \mathscr{D}_A))$ with a normal distribution that closely matches the shape of the original distribution around its mode. However, computing $\log(p(\theta \mid \mathscr{D}_A))$ is usually intractable. Assume $\mathscr{D}_A = \{\mathbf{x}_A^{(i)}, y_A^{(i)}\}_{i \in [|1, N_A|]}$ with the samples being independent and identically distributed (iid). We will denote $p(\mathbf{x}_A)$

and $p(y_A)$ as their respective common distributions. We will use the Bayes rule to invert the probabilities and express the intractable posterior with the likelihood and the prior:

$$\begin{aligned} p(\theta \mid \mathscr{D}_A) &\propto p(\mathscr{D}_A \mid \theta) p(\theta) \\ &\propto p\left(y_A \mid \theta, \mathbf{x}_A\right)^{N_A} p(\theta) \quad \text{(iid)} \end{aligned} \tag{4.7}$$

Going back to the log space:

$$\log p(\theta \mid \mathscr{D}_A) = N_A \log p\left(y_A \mid \theta, \mathbf{x}_A\right) + \log p(\theta) + C_1 \tag{4.8}$$

where $C_1 \in \mathbb{R}$ is a constant with respect to θ.

Assume the parameters follow a zero-mean isometric Gaussian prior, i.e. $\theta \sim \mathscr{N}\left(\mathbf{0}, \frac{1}{\lambda} \mathrm{I}_p\right)$ where $\lambda \in \mathbb{R}_+$ and $\mathrm{I}_p \in \mathscr{M}_{p \times p}(\mathbb{R})$ is the identity matrix. Therefore, we have:

$$\log p(\theta) \propto \lambda \theta^\top \theta \tag{4.9}$$

Hence:

$$\frac{\partial^2 \log_p(\theta)}{\partial \theta^2} = 2\lambda \mathrm{J} \tag{4.10}$$

where J is a matrix full of ones.

Now, Equation 4.6 gives:

$$\Sigma^{-1} = N_A \left(\frac{\partial^2 (\log((y_A \mid \theta, \mathbf{x}_A)}{\partial \theta^2} \right) \Bigg|_{\theta_A^\star} + 2\lambda \mathrm{J} \tag{4.11}$$

If we further assume that $\lambda \mathrm{J} \ll N_A \left(\frac{\partial^2 (\log((y_A \mid \theta, \mathbf{x}_A)}{\partial \theta^2} \right) \Big|_{\theta_A^\star}$ (which we can consider true if we have a high enough value for N_A), we can infer:

$$\Sigma^{-1} \approx \left(\frac{\partial^2 (\log(p(\mathscr{D}_A \mid \theta)))}{\partial \theta^2} \right) \Bigg|_{\theta_A^\star} \tag{4.12}$$

Let's put this aside for a moment and let's show that the variance of our Laplace approximation is actually related to the observed Fisher information matrix $\widetilde{\mathscr{F}_A}$ at $\theta_A^\star$. Therefore, let's first derive an expression of the Fisher information matrix $\mathscr{F}_A$:

$$\mathscr{F}_A = \mathrm{Var}_{p(\theta \mid \mathscr{D})} \left[\left(\frac{\partial (\log(p(\mathscr{D}_A \mid \theta)))}{\partial \theta} \right) \Bigg|_{\theta_A^\star} \right] \tag{4.13}$$

Let the score vector $\mathbf{S}_{\theta_A^\star} = \left(\frac{\partial(\log(p(\mathscr{D}_A|\theta)))}{\partial\theta}\right)\Big|_{\theta_A^\star}$. Using the König-Huygens theorem, we get:

$$\mathscr{F}_A = \mathbb{E}_{p(\theta|\mathscr{D})}\left[\mathbf{S}_{\theta_A^\star}\mathbf{S}_{\theta_A^\star}^T\right] - \left(\mathbb{E}_{p(\theta|\mathscr{D})}\left[\mathbf{S}_{\theta_A^\star}\right]\right)\left(\mathbb{E}_{p(\theta|\mathscr{D})}\left[\mathbf{S}_{\theta_A^\star}\right]\right)^T \tag{4.14}$$

Assuming the regularity conditions of the Leibniz integral rule for differentiation under the integral sign hold, and using $\mathbb{E}_{p(\theta|\mathscr{D})}p(\theta \mid \mathscr{D}) = 1$, we get that the Fisher information matrix is uniquely defined by the second moment of the score vector:

$$\mathscr{F}_A = \mathbb{E}_{p(\theta|\mathscr{D})}\left[\left(\left(\frac{\partial(\log(p(\mathscr{D}_A \mid \theta)))}{\partial\theta}\right)\left(\frac{\partial(\log(p(\mathscr{D}_A \mid \theta)))}{\partial\theta}\right)^\top\right)\Big|_{\theta_A^\star}\right] \tag{4.15}$$

Moreover, under a few extra regularity conditions that we will assume satisfied here, we have:

$$\mathscr{F}_A = -\mathbb{E}_{p(\theta|\mathscr{D})}\left[\frac{\partial^2(\log(p(\mathscr{D}_A \mid \theta)))}{\partial\theta^2}\Big|_{\theta_A^\star}\right] \tag{4.16}$$

where $\theta_A^\star$ is the vector that contains the optimal weights for the model for the training set for task A.

Now, as a reminder, the observed Fisher information matrix at $\theta_A^\star$ is derived from the Fisher information matrix by considering the random variable θ as a fixed observation. Thus, we can now infer the observed Fisher information matrix on θ, denoted $\widetilde{\mathscr{F}_A}$:

$$\widetilde{\mathscr{F}_A} = -\left(\frac{\partial^2(\log(p(\mathscr{D}_A \mid \theta)))}{\partial\theta^2}\right)\Big|_{\theta_A^\star} \tag{4.17}$$

Now we can go back to Equation 4.12 and observe that $\Sigma = \widetilde{\mathscr{F}_A}^{-1}$. Therefore, we can identify the invert of the observed Fisher information matrix in Equation 4.5:

$$p(\theta \mid \mathscr{D}_A) \sim \mathscr{N}\left(\theta_A^\star,\ \widetilde{\mathscr{F}_A}^{-1}\right) \tag{4.18}$$

With our new approximation $p(\theta \mid \mathscr{D}_A) \sim \mathscr{N}\left(\theta_A^\star, \widetilde{\mathscr{F}_A}^{-1}\right)$, we can now simplify Equation 4.2:

$$\log(p(\theta \mid \mathscr{D}_A,\ \mathscr{D}_B)) = \log(p(\mathscr{D}_B \mid \theta)) + \frac{1}{2}(\theta - \theta_A^\star)^\top \widetilde{\mathscr{F}_A}(\theta - \theta_A^\star) + C_2 \tag{4.19}$$

where $C_2 \in \mathbb{R}$ is a constant with respect to θ.

Apart, let's define R_{EWC} such that:

$$R_{\text{EWC}}(\theta) = \frac{1}{2}(\theta - \theta_A^\star)^\top \widetilde{\mathscr{F}_A}(\theta - \theta_A^\star) \tag{4.20}$$

Now, we can infer the loss function for EWC:

$$\mathscr{L}_{\text{EWC}} = \mathscr{L}_B + \lambda R_{\text{EWC}}(\theta) \tag{4.21}$$

where:

- $\mathscr{L}_{\text{EWC}}$ is the complete loss function to fine-tune using EWC
- $\mathscr{L}_B$ is the original loss used to learn task B
- $\lambda \in \mathbb{R}_+^*$ is a hyperparameter we introduced to have more control over the importance of task B compared to task A ($\lambda = 1$ if we strictly derive the result from Equation 4.19).

4.1.2 Implementation

To implement EWC, we need to estimate the first two moments of $\mathscr{N}\left(\theta_A^\star, \widetilde{\mathscr{F}_A}^{-1}\right)$. We will refer to them as the EWC parameters in this thesis [1]. For the mean $\theta_A^\star$, we can store the value of the model weights after training for task A. For the variance, we need to compute the observed Fisher information matrix (see Equation 4.15), defined by a first-order gradient. To compute the gradient, we will use the auto-diff capabilities of PyTorch to compute the first-order derivative of the log-likelihood with respect to task A by simply running inference on the dataset A. Because PyTorch operates on a batch level, let's derive the equation for $\widetilde{\mathscr{F}_A}$ with respect to batches of samples.

Let's denote:

- N_{batch} is the number of batches for the dataset $\mathscr{D}_A$
- For all $i \in [|1, N_{\text{batch}}|]$, A_i is the i-th batch for the dataset $\mathscr{D}_A$
- $\mathscr{D}_A = \bigcup_{i=1}^{N_{\text{batch}}} \mathscr{D}_{A,i}$
- $\mathbf{x}_{A,i}^{(n)}$ the n-th example in the i-th batch $\mathscr{D}_{A,i}$ and $y_{A,i}^{(n)}$ its associated label

[1] It is interesting to note that having access to the pre-computed EWC parameters for the previous task would have been enough. Assuming EWC is efficient, it would then be good practice to release them alongside the future foundation large pre-trained model's weights.

Assuming the samples are independent and identically distributed, we get:

$$\widetilde{\mathscr{F}_A} = \sum_{i=1}^{N_{\text{batch}}} \sum_{n=1}^{|\mathscr{D}_{A,i}|} \left[\left(\left(\frac{\partial(\log(p(y_{A,i}^{(n)} \mid \mathbf{x}_{A,i},\ \theta)))}{\partial \theta} \right) \left(\frac{\partial(\log(p(y_{A,i}^{(n)} \mid \mathbf{x}_{A,i},\ \theta)))}{\partial \theta} \right)^{\top} \right) \Bigg|_{\theta_A^\star} \right] \tag{4.22}$$

Going back to Equation 4.20, we have two matrix operations with a total time complexity of $\mathscr{O}(P^2)$ where P is the total number of parameters. Another problem comes with the memory complexity of storing $\widetilde{\mathscr{F}_A}$ in $\mathscr{O}(P^2)$. Thus, Kirkpatrick et al. decided to further approximate $\widetilde{\mathscr{F}_A}$ by its diagonal matrix, which we will denote $\text{diag}\{F_1^{(A)}, \ldots, F_P^{(A)}\}$ [39]. Therefore:

$$\forall p \in [|1,P|], F_p^{(A)} = \left[\left(\frac{\partial(\log(p(\mathscr{D}_A \mid \theta)))}{\partial \theta_p} \right) \Bigg|_{\theta_A^\star} \right]^2 \tag{4.23}$$

Using Equation 4.22, we can also infer the batch-wise expression of $\text{diag}\{F_1^{(A)}, \ldots, F_P^{(A)}\}$:

$$\forall p \in [|1,P|], F_p^{(A)} = \sum_{i=1}^{N_{\text{batch}}} \sum_{n=1}^{|\mathscr{D}_{A,i}|} \left[\left(\frac{\partial(\log(p(y_{A,i}^{(n)} \mid \mathbf{x}_{A,i},\ \theta)))}{\partial \theta_p} \right) \Bigg|_{\theta_A^\star} \right]^2 \tag{4.24}$$

Finally, Equation 4.20 can be further simplified as:

$$\begin{aligned} \mathscr{L}_{\text{EWC}} &= \mathscr{L}_B + \lambda R_{\text{EWC}}(\theta) \\ &\approx \mathscr{L}_B + \frac{\lambda}{2} \sum_{p=1}^{P} F_p^{(A)} \left(\theta_p - \theta_{A,p}^\star\right)^2 \end{aligned} \tag{4.25}$$

Note that computing R_{EWC} in Equation 4.25 necessitates both a time complexity and a space complexity of $\mathscr{O}(P)$, which is a significant improvement from the previous complexities in $\mathscr{O}(P^2)$.

4.2 Task Alignment Consolidation

While EWC tackles forgetting by applying regularization from the parameter space, we want to regularize the model directly from the prediction space because we hypothesize that considering the fully generated sequences is more meaningful. An additional challenge is that we might not have the dataset used for the original task. For the last issue, Li et al. proposed *Learning without Forgetting* (LwF) and experimented on it on computer vision classification tasks [42]. Task Alignment Consolidation (TAC) is a novel method we have developed that aims at adapting LwF for sequence-to-sequence ASR multitask models like Whisper.

4.2.1 Definition

TAC aims at preserving the non-English weights during training by making the model *consistently bad* at predicting, e.g., French from English audio. Let N_{task} be the number of non-target tasks we want our model to remember after training. For all $n \in [|1, N_{\text{task}}|]$, let $\mathscr{R}_n$ be the penalty term associated with the n-th task, which is a similarity measure between the original model's output and the current trained one. We can then define the TAC penalty term R_{TAC}:

$$R_{\text{TAC}} = \frac{1}{N_{\text{task}}} \sum_{n=1}^{N_{\text{task}}} \mathscr{R}_n \tag{4.26}$$

Then, we can infer the final TAC loss:

$$\begin{aligned} \mathscr{L}_{\text{TAC}} &= \mathscr{L} + \gamma R_{\text{TAC}} \\ &= \mathscr{L} + \frac{\gamma}{N_{\text{task}}} \sum_{n=1}^{N_{\text{task}}} \mathscr{R}_n \end{aligned} \tag{4.27}$$

where:

- $\mathscr{L}$ is the original loss of interest i.e. the loss for the new task
- $\gamma \in \mathbb{R}_+^*$ the amount of regularization for TAC
- R_{TAC} is the negative similarity between the original model and the current trained sequences.

Examples of relevant negative similarity functions are the negative cross-entropy and the KL divergence - both computed in a teacher-forced fashion.

4.2.2 Implementation

Whisper takes two special tokens as inputs before the audio features: `<|language|>` and `<|task|>`. Thus, we can prompt Whisper with these tokens to encourage Whisper to generate outputs for the tasks and the languages we want to preserve. [2] Although the model will likely output gibberish transcriptions - which we will refer to as *pseudo-transcriptions* - we hypothesize that the generated sequences will still align with Whisper's out-of-task capabilities. For instance, we hypothesize that, if prompted to transcribe French from English audio, Whisper will output French words that sound similar to the reference English ones. Thus, fine-tuning with the TAC regularization should force the model to keep the non-English weights primarily untouched and *a fortiori* prevent forgetting. The generic procedure for training using TAC is shown in Figure 4.2.

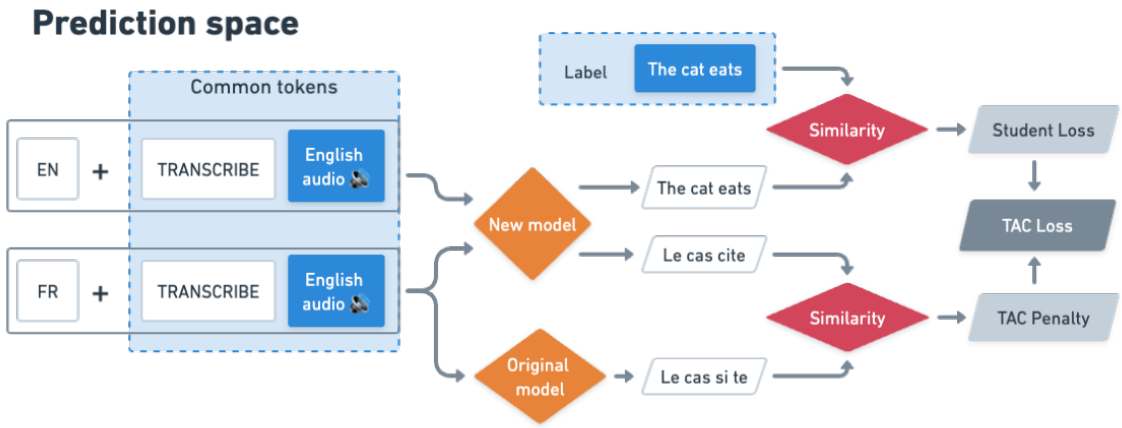


Fig. 4.2 TAC training strategy for fine-tuning Whisper on an English transcription task while tackling forgetting on French transcription.

Looking back on both EWC and TAC, we want to highlight that the two methods are based on opposite assumptions. While EWC assumes the model's nodes related to different languages to be distinct, TAC is built on the premise that they are entangled to a certain extent. Consequently, we expect that at most one of both continual learning will work.

[2] Initially, we wanted to align the English→X translations instead of the X transcriptions where X is the language to preserve as the outputs would not be gibberish anymore. However, at the time of writing, Whisper only supports X→English translation.

Chapter 5

Experimental setup

Now that we have all the theoretical background explained, we can shift our attention to the preliminary decisions we have to make before running our experiments. First, we will introduce all the training and evaluation datasets that will be used for this project. Second, we will define our evaluation framework. Last but not least, we will evaluate the pre-trained Whisper models - which we will refer to as *vanilla* - to get our baseline figures.

5.1 Datasets

All the datasets at hand have their reference text upper-cased. Knowing that Whisper wasn't pre-trained on fully upper-cased text, we hypothesized and empirically confirmed that Whisper couldn't learn with this formatting. Consequently, we decided to lowercase all the references of the datasets at hand.

5.1.1 Target datasets

The data has been downloaded using HuggingFace's `datasets` [40] to make the experiments easily reproducible. We filtered out all the examples with audio lengths not longer than 30 seconds and those with empty references. Each training dataset will be divided into 3 splits: train, validation, and test. The training split is a subset of the available data used to train a machine learning model. The validation split is a subset of data used to monitor the metric of interest (e.g. the WER) during training and to perform hyperparameter tuning. The test set is an independent subset of held-out data that is used to evaluate the final performance of the trained model.

LibriSpeech clean

The first target dataset is LibriSpeech clean [53]. It is a widely used ASR dataset that consists of English speech recordings derived from audiobooks in the LibriVox project. The dataset covers a diverse range of speakers, reading materials, and acoustic conditions. For training, we will use the 100h-long training split. Details about the dataset can be found in Table 5.1.

Split	# examples	Total duration (h)
Train	28539	100.6
Validation	2703	5.4
Test	2620	5.4

Table 5.1 Details about the LibriSpeech clean dataset.

The Augmented Multi-party Interaction dataset (AMI)

The Augmented Multi-party Interaction (AMI) is a 100-hour corpus of meeting recordings [9]. This report will only consider the IHM split which so consists of audio recorded with microphones close to the speakers. Yet, the AMI-IHM (referred to as AMI for the rest of this thesis) dataset is still much noisier than LibriSpeech and contains plenty of hesitations and repetitions. However, because our end goal is to work on the ALTA dataset which is also about conversational speech, we hypothesize that working on AMI will provide a good indication of what level of performance we will be able to achieve for ALTA. Therefore, a particular focus will be shown on AMI compared to LibriSpeech. Details about the dataset can be found in Table 5.2.

Split	# examples	Total duration (h)
Train	108492	77.86
Validation	12643	8.68
Test	12643	5.70

Table 5.2 Details about the AMI-IHM 100h dataset.

5.1.2 Evaluation datasets

To evaluate Whisper on the different tasks of interest, we created three dataset groups to structure the way we will present our results: ESB diagnostic custom (ESBDC), MultiLingual LibriSpeech dataset (MLS) [56], and Forgetting Assessment Dataset (FAD). A partial summary is shown in Table 5.3 while full details can be found in Appendix A.

	# datasets	Multilingual	Description
ESBDC	8		Used for out-of-distribution evaluation
MLS	8	✓	Used for out-of-task evaluation
FAD	3	✓	AMI, TED-LIUM, and MLS French

Table 5.3 Evaluation dataset groups used in this thesis.

5.2 Text normalization strategy

5.2.1 Normalized evaluation

There is a wide range of textual representations for a same spoken content. For instance, it is unclear whether a number should be written with digits or with letters, or whether we should always use contractions in transcriptions. While the raw WER penalizes ASR models for innocuous differences, we want to mitigate the impact of linguistic variations to have a more robust metric in practice. Text normalization proposes to solve this issue by standardizing these diverse forms. For this purpose, the Whisper paper [57] introduced two normalizers. First, basic normalization removing casing and punctuation. The second normalizer is specific to English and introduces various rule-based edits (hesitation removal, number normalization, British to American spelling...). In this thesis, the English normalizer will be used for all English transcription tasks while the basic one will be used for non-English transcriptions.

While lower WERs are desirable, generating easy-to-read text is equally important as our work ultimately aims at transcribing audio into text destined to be read by humans. Thus, we should preserve casing and punctuation if available. While all our target sets are lowercased (see subsection 5.1.1), we might be interested in taking advantage of the casing and punctuation of the teacher outputs (see subsection 2.2.5) during distillation as the prosodic information contains a valuable signal for training [37]. All the possible pre and post-normalization strategies are compared in Table 5.4. This table motivated our choice to train our models on non-normalized transcriptions but to evaluate them with normalization.

5.2.2 Teacher normalization for knowledge distillation

Normalization is crucial for knowledge distillation as the formatting learned by the pre-trained Whisper models might not be the same as the one of the target dataset. For example, while the vanilla Whisper is pre-trained to use casing and punctuation, all references are lower-cased and without punctuation. At evaluation time, this issue is negligible because of normalization subsection 5.2.1. However, we hypothesize that formatting will be a problem

Train	Eval	Pros	Cons
None	None	* Predict casing/punctuation * One logic for train/eval	* Prediction WER higher
None	**Normalize**	* Predict casing/punctuation * WER for predictions is lower	* Different logic for train/eval
Normalize	Normalize	* One logic for train/eval * WER for predictions is lower	* No prosodic signal for train * No casing/punctuation

Table 5.4 Comparison between the different normalization strategies available. The **highlighted** row is the normalization strategy we will use.

for knowledge distillation: if we use both the reference and the raw teacher outputs during knowledge distillation, we may confuse the student as it would get taught two forms of formatting simultaneously. Finally, we noticed that most teacher predictions started with a whitespace. We also hypothesize that this extra whitespace will be detrimental to distillation.

To solve these issues, we propose to remove all punctuation (except apostrophes as they are present in the references) and trailing whitespaces from the teacher outputs. An example of teacher post-processing is shown in Table 5.5.

Reference	"yeah i would suggest that too"
Teacher output	
Raw	" Yeah, I would suggest that too."
Casing/punctuation removed	" yeah i would suggest that too"
Trailing whitespaces removed	"Yeah, I would suggest that too."
Casing/punctuation/trailing whitespaces removed	"yeah i would suggest that too"

Table 5.5 Example of the effect of post-processing on the teacher's generated predictions.

Invert text normalization is a possible source of increased WER. In particular, we noticed that the acronyms in AMI's references are always transcribed with each letter being separated by a whitespace followed by a dot while the vanilla Whisper would use the usual writing style. This phenomenon is illustrated in Table 5.6. While this issue wasn't addressed in this thesis, we believe that aligning the teacher formatting with the target dataset can improve the resulting WER significantly.

Reference	"i don't know put all the p. h. d. students together all the professors together that sort of thing"
Teacher output	" I don't know, put all the PhD students together, all the professors together, and that sort of thing."

Table 5.6 Example of discrepancy for inverse text normalization (in red).

5.3 Decoding strategy

5.3.1 Prompting strategy

Whisper can automatically predict the task of interest based on the nature of the input. For example, if the model is given audio to transcribe, the model will automatically infer the language that should be used. Nonetheless, we can also force Whisper to predict following a specific task by appending the `<|language|>` special token followed by the `<|task|>` right after the start-of-transcript token. We will refer to this process as *forced prompting*. In this thesis, we decided to always use forced prompting before decoding.

5.3.2 Generation strategy

Maximum sequence length

As a generative model, Whisper will keep predicting new tokens until it generates an end-of-sentence token. This is a problem when Whisper hallucinates, as it can confidently predict a very long output sequence of gibberish tokens. For this thesis, we empirically set the maximum length sequence to 255 tokens to limit the output size in case the model get stuck in a repetition loop.

Decoding strategy

While a larger beam size allows for a broader exploration of possible hypotheses, it can also lead to potentially suboptimal paths with higher WERs. Thus, we studied the impact of the beam size for decoding with Whisper.

Looking at the results in Figure 5.1, we observe that the surge in WER is mainly due to the increased insertion errors. Digging into the predictions with high WER, we showed in Table 5.7 that Whisper is more likely to be stuck in repetition loops for higher values of beam sizes.

idx	label	k=1		k=5	
		pred	WER (%)	pred	WER (%)
117	yeah	yeah	0	yeah yeah ... yeah [x222]	221
152	yeah	you	100	haha haha ... haha [x255]	255
156	yeah	hahaha	100	ha ha ... ha [x255]	255

Table 5.7 Example of hallucinations when using k-beam search decoding with the vanilla Whisper `tiny` on the AMI 100h test split. Note that 255 is the maximal generation length set during evaluation.

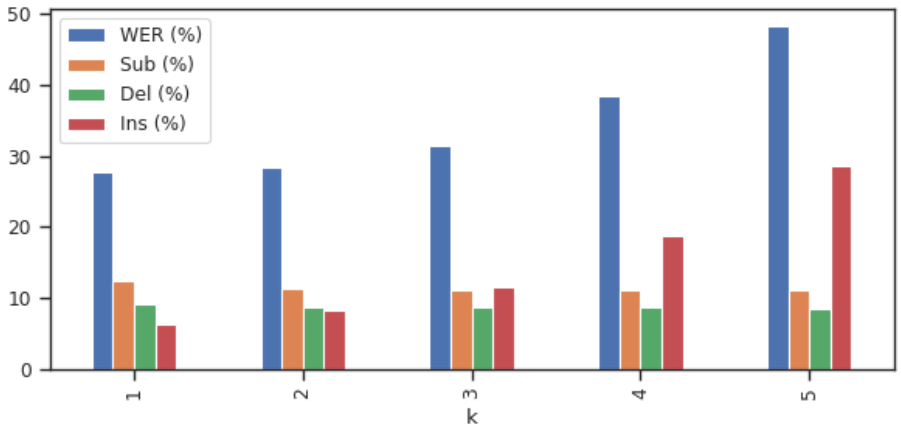


Fig. 5.1 Evolution of the WER metrics for the vanilla Whisper `tiny` decoded with K-beam search on AMI test with respect to the beam size K.

Repetitions appear to be common to both greedy search and K-beam search [66, 68]. Holtzman et al. have shown that human-like texts are more "surprising" as the next word in a sentence is quite often not the one with the highest conditional probability. The same authors have also demonstrated that the probability of a repeated phrase increases with each repetition, creating a positive feedback loop. We confirmed that this phenomenon affected Whisper `tiny` as well in Figure 5.2.

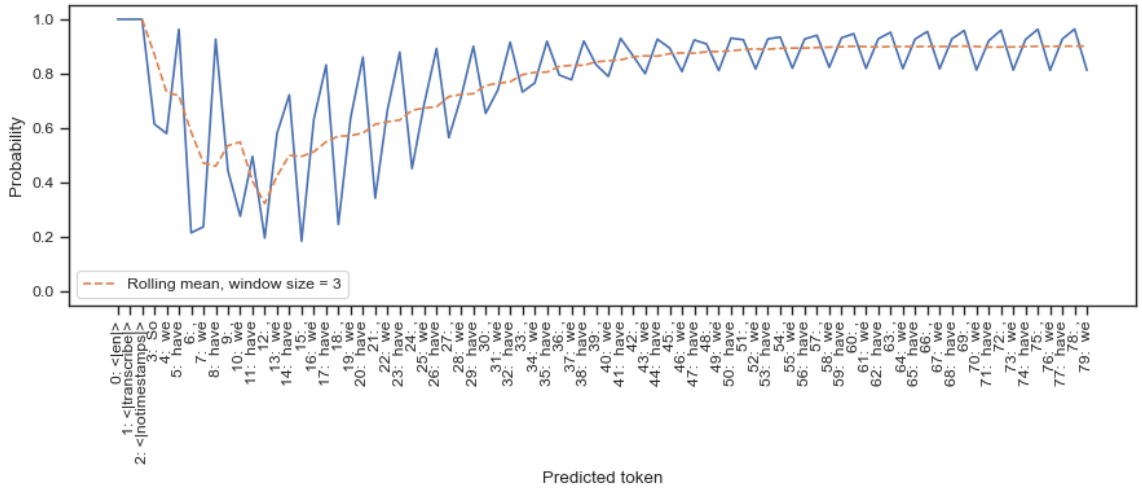


Fig. 5.2 Evolution of the probability of the next predicted token for an arbitrary example from AMI validation. The reference is `"so we have oh no what's that"`.

We compared all the mentioned decoding strategies using the WER on the test split of AMI. Results are shown in Table 5.8, with hyperparameters chosen according to the recommendations in the literature.

	$\text{WER}_{(\%)}$	$\text{del}_{(\%)}$	$\text{sub}_{(\%)}$	$\text{ins}_{(\%)}$
Greedy search	**27.73**	9.16	12.38	6.19
K-beam search ($K = 3$)	31.42	8.69	11.12	11.61
K-beam search ($K = 5$)	48.33	**8.57**	**11.04**	28.72
Vanilla sampling	54.32	14.2	23.53	16.59
Top-k sampling ($k = 40,\ \tau = 0.7$)	31.57	10.3	15.21	6.06
Top-p sampling ($p = 0.92$)	41.45	11.7	18.66	11.09
Top-k + top-p sampling ($k = 40,\ p = 0.92,\ \tau = 0.7$)	30.04	9.82	14.14	6.08

Table 5.8 Comparison of different decoding strategies with the vanilla Whisper `tiny` evaluated on the AMI 100h test split. The best metrics per dataset are **highlighted**.

Looking at Table 5.8, greedy search is the best decoding strategy. For this reason and for its low time complexity, we decided to use greedy search for the rest of this thesis.

5.4 Whisper vanilla performance

5.4.1 Results

We decided to use the Whisper implementation from HuggingFace's `transformers` [71] as we measured an 8-time speed-up compared to OpenAI's implementation. Moreover, flash attention [16] and mixed precision [51] were used to accelerate inference. So not only do these choices allow faster training, but they will also reduce the inference costs of the deployed ASR model for ALTA.

The details of the WER metrics for the vanilla Whisper models evaluated on the test slit of AMI are shown in Table 5.9. Note that as the `large-v2` model could not fit within the available GPU memory on our hardware, we decided to discard it from this work.

	Parameters (M)	WER (%)	Del (%)	Sub (%)	Ins (%)
`tiny`	39	27.74	9.15	12.38	6.21
`base`	74	22.41	8.99	9.08	4.33
`small`	244	19.05	8.98	6.69	3.38
`medium`	769	16.62	8.62	5.74	2.25

Table 5.9 Evolution of the WER metrics for the different vanilla Whisper models on the AMI test split.

As expected, the larger the model, the lower the WER. To maximize the efficiency of distillation, we plan to distil the most performant Whisper model into the smallest one. Hence, we decided to distil `medium` into `tiny` (see Figure 5.3).

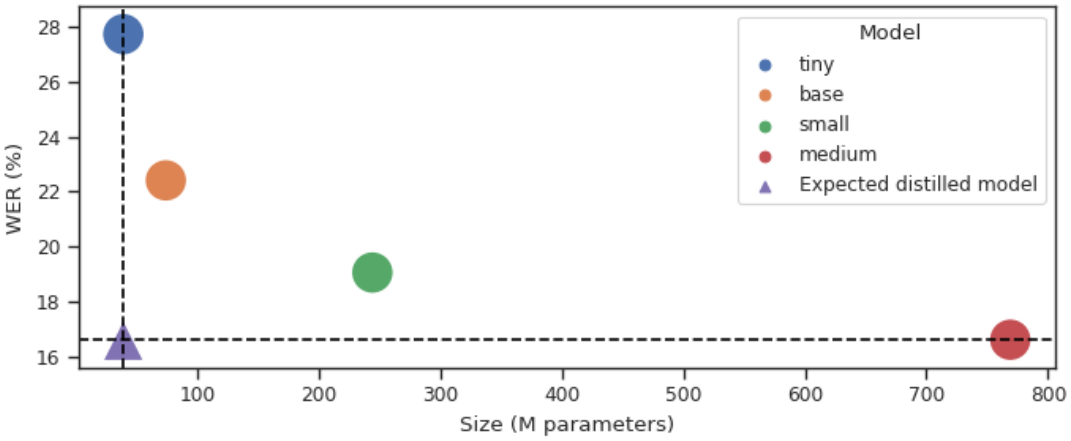


Fig. 5.3 Evolution of the WER of the vanilla Whisper models with respect to their model size. The models were evaluated on the AMI 100h test split.

We are now interested in comparing the robustness of both the student and the teacher on the other English ASR datasets from the ESBDC dataset group. Results are shown in Table 5.10, where $\Delta_{\text{rel}}(\text{WER})$ is the relative WER reduction.

Dataset	Vanilla `tiny`	Vanilla `medium`	$\Delta_{\text{rel}}(\text{WER})$ (%)
librispeech clean	7.54	**2.88**	61.74
librispeech other	16.88	**6.04**	64.19
common voice	17.19	**3.26**	81.06
voxpopuli	8.06	**4.96**	38.46
tedlium	5.18	**2.64**	49.04
gigaspeech	8.29	**4.44**	46.40
spgispeech	5.74	**2.49**	56.66
earnings22	18.54	**10.94**	40.99
ami	27.74	**16.61**	40.13
Average	12.79	**6.03**	52.88

Table 5.10 WER (%) comparison between the vanilla `tiny` and `medium` Whisper models evaluated on the ESBDC dataset. The best metrics per dataset are **highlighted**.

In Table 5.10, we can see that the vanilla `medium` Whisper model performs significantly better compared to the `tiny` vanilla model, with an average relative WER reduction of 52% on the ESBDC dataset group. This further confirms that `medium` is a promising teacher for distilling `tiny`.

5.4.2 Analysis

As we expect the errors of the teacher model to propagate to the student, it is worth analyzing the nature of these different mistakes for the target distribution.

Disfluency removal

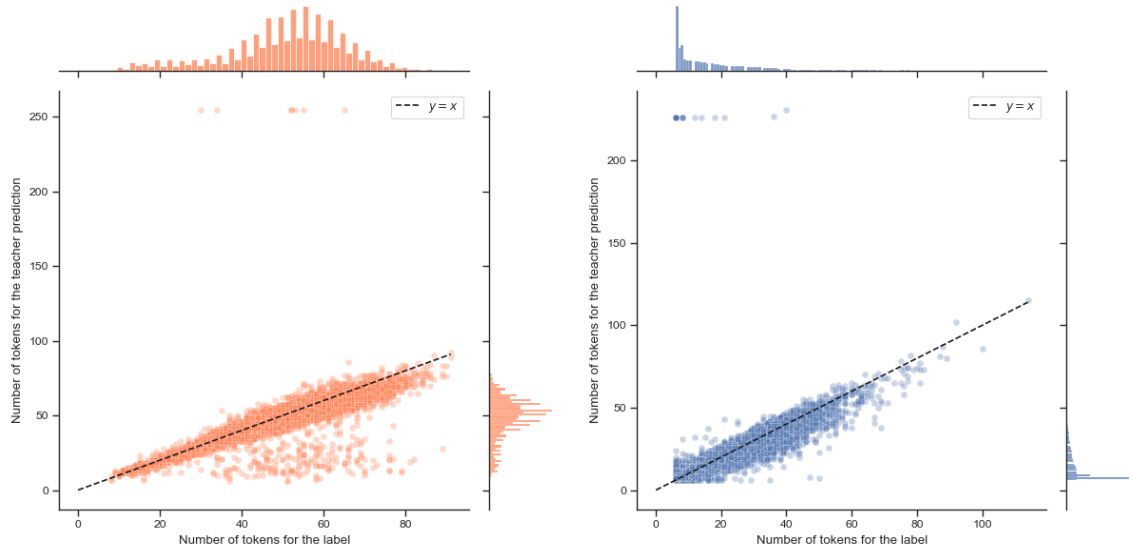
The reference texts for AMI contain disfluencies such as repetitions and hesitations. Although hesitations will get removed by normalization before WER scoring (see subsection 5.2.1), we would like to preserve them as they are crucial in assessing the fluency of a candidate for ALTA's use case. On the other hand, the repetitions will be more of a problem during unsupervised distillation as the student will have no way to learn that it should preserve them. Examples of removed disfluencies are shown in Table 5.11.

References	Vanilla `medium` predictions
so um yeah you got a general f uh f the functions of the device uh for a d. v. d. player or uh so um the pl yeah um f for uh playing uh reverse uh	so you've got the general functions of the device for a dvd player for playing reverse data
uh uh o one of uh out of s uh sev um it's uh easier to uh to do the th things that are like that on a computer	it's easier to do things like that on the computer

Table 5.11 Examples of disfluencies removed (in red) by Whisper. Examples are taken from validation split of AMI.

Repetition-based hallucinations

We are interested in investigating how often and how strongly repetition-based hallucinations occur with the teacher Whisper. First, we decided to inspect the difference in number of output tokens with the ground truth. The distribution of the number of tokens is shown in Figure 5.4.



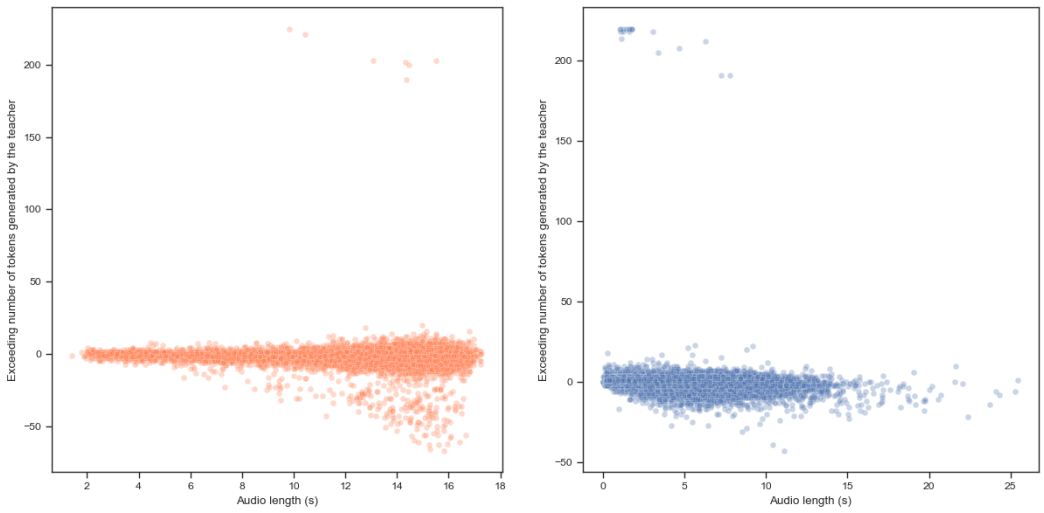
(a) LibriSpeech **(b)** AMI

Fig. 5.4 Evolution of the number of tokens in the `medium` Whisper's outputs with respect to the number of tokens in the references.

We can observe that most predictions have roughly the same number of tokens as the labels. However, it seems that a few predictions have a significantly higher number of tokens than they should have. Note that this phenomenon is very rare with less than 0.01% rows of LibriSpeech and AMI train having more than 50 excess tokens. Yet, repetitions are often generated until the model reaches the generation length limit. And because the WER is

upper-bounded by the number of words, even a few hallucinations can significantly impact the performance of our model.

Finally, we looked at the relationship between the number of exceeding tokens and the audio length. For LibriSpeech, repetition-based hallucinations tend to occur for audio longer than 8 seconds, but no obvious trend could be observed. On the other hand, hallucinations on AMI are more frequent for audio shorter than 2 seconds. Results are shown in Figure 5.5.



(a) LibriSpeech **(b)** AMI

Fig. 5.5 Evolution of the exceeding number of tokens generated by the teacher with respect to the audio length.

Non-focused transcriptions

Whisper tends to transcribe speech from people talking in the background. This is not a problem for LibriSpeech, which consists of people reading audiobook passages in a noise-free environment [53]. However, it is problematic for the AMI corpus, which consists of noisy meeting recordings [9]. While we hypothesize fine-tuning should easily teach Whisper to focus on the main speaker (*a fortiori* decrease the insertion word error rate), we believe we can hardly tackle this problem with unsupervised distillation i.e. without the references at hand. We will refer to this issue as *non-focused transcriptions*. A few non-focused transcriptions from the validation split of AMI 100h are shown in Table 5.12. Finally, to wrap things up, the impact of the errors from the vanilla Whisper on the WER metrics is also shown in Table 5.13.

References	**Vanilla `medium` teacher predictions**
in a two person office	In a two person office? No, in a three person office.
no no no no let's do the lounge corner	No, no, no, no. Let's do the lounge corner. The lounge is in the free room.
i don't know but it was a good performance creepy	I don't know, but it was a good performance. Yes, very. Creepy.

Table 5.12 Examples of non-focused transcriptions (in red) from the vanilla `medium` Whisper on the validation split of AMI.

Type of misalignment	Del	Sub	Ins
Disfluencies	✓		
Repetition-based hallucinations			✓
Non-focused transcriptions			✓

Table 5.13 Impact of the errors from the vanilla Whisper on the WER metrics.

Chapter 6

Results and discussion

This chapter is about the results obtained from knowledge distillation and continual learning and will build up on the theoretical background presented in the previous chapters. In the first place, we will investigate supervised fine-tuning. Second, we will investigate different unsupervised distillation methods. Third, we will look at different methods to implement continual learning.

6.1 Supervised fine-tuning

6.1.1 Results

We fine-tuned the vanilla `tiny` Whisper model on the 100h train split of LibriSpeech clean. We used a batch size of 32, `AdamW` [44] as our optimizer with a scheduler with a warmup during the first 100 steps followed by a constant learning rate. After trying learning rates in the range {`1e-5`, `5e-5`, `1e-4`, `5e-4`}, we obtained the optimal learning rate value of `1e-5`. All models were trained for 5 epochs. We have also investigated three strategies for freezing the model parameters: not freezing any layer, freezing the encoder, and freezing the decoder. Freezing the encoder achieved the lowest WER, thus we decided to opt for this strategy during fine-tuning. Except if stated otherwise, this training configuration will be used for the rest of this chapter. Fine-tuning results are shown in Table 6.1, where $\Delta_{\text{rel}}(\text{WER})$ is the relative WER reduction.

Initially, we had planned to use LibriSpeech clean as the target distribution for knowledge distillation. However, Table 6.1 shows that the `tiny` vanilla model already performs very well on this dataset with a WER of 7.54%. Assuming we manage to reach the `medium` performance after distillation, we would end up with a WER of 2.88 i.e. an absolute improvement of 4.66 points. Consequently, it is sensible to pick AMI instead as it has the highest absolute

	Vanilla `tiny`	Fine-tuned `tiny`	Δ_{rel}(WER) (%)
WER (%)	7.54	**7.14**	5.31
Ins (%)	1.21	**1.06**	12.40
Sub (%)	5.47	**5.23**	4.39
Del (%)	0.86	**0.85**	1.16

Table 6.1 WER metrics (%) comparison between the vanilla and fine-tuned `tiny` Whisper models evaluated on the LibriSpeech clean test dataset. Fine-tuning was performed on the train split of LibriSpeech clean. The best metrics per dataset are **highlighted**.

WER reduction (11.13 points) between `tiny` and `medium` among all the datasets evaluated in Table 5.10.

Following our previous observations, we fine-tuned Whisper `tiny` on the AMI train split. The model was trained for 1 epoch. The same training hyperparameters were used compared to LibriSpeech, with the exception of the increased number of warmup steps of 600. Fine-tuning results are shown in Table 6.2 and a relative WER reduction of 25.78% is demonstrated.

	Vanilla `tiny`	Fine-tuned `tiny`	Δ_{rel} (%)
WER (%)	27.74	**20.59**	25.78
Ins (%)	6.19	**4.65**	24.88
Sub (%)	12.38	**10.46**	15.51
Del (%)	9.16	**5.48**	40.18

Table 6.2 WER metrics (%) comparison between the vanilla and fine-tuned `tiny` Whisper models evaluated on the AMI test dataset. Fine-tuning was performed on the train split of AMI. The best metrics per dataset are **highlighted**.

6.1.2 Analysis

As discussed in chapter 4, we expect Whisper to forget about the non-target distributions during fine-tuning on AMI. As we hypothesize distillation will have a similar impact on forgetting, we will consider the previously fine-tuned model first. To begin with, we will analyze forgetting when transcribing English speech from other datasets than the target one (out-of-distribution). Afterward, we will do the same for transcribing non-English speech (out-of-task).

Out-of-distribution English datasets

All measures of out-of-distribution forgetting relative to fine-tuning have been performed on the ESBDC dataset group. The WER robustness for out-of-distribution English datasets of the `tiny` model fine-tuned on AMI can be found in Table 6.3 where $\Delta_{\text{rel}}(\text{WER})$ is the relative WER reduction.

Dataset	Vanilla `tiny`	Fine-tuned `tiny`	$\Delta_{\text{rel}}(\text{WER})$ (%)
In-distribution			
AMI	27.74	**20.59**	25.78
Out-of-distribution			
librispeech clean	**7.54**	12.27	-62.82
librispeech other	**16.88**	24.52	-45.30
common voice	**17.19**	26.25	-52.71
voxpopuli	**8.06**	12.47	-54.80
tedlium	**5.16**	6.49	-25.78
gigaspeech	**8.29**	11.04	-33.15
spgispeech	**5.74**	9.24	-61.06
earnings22	**18.54**	19.95	-7.60
Out-of-distribution average	**10.93**	15.28	-42.85
Average	**12.79**	15.87	-24.04

Table 6.3 WER (%) comparison between the vanilla and fine-tuned `tiny` Whisper models evaluated on the ESBDC dataset. Fine-tuning was performed on the train split of AMI. The best metrics per dataset are **highlighted**.

The fine-tuned Whisper's performance on the out-of-distribution datasets significanlty worsened with an average relative WER increase of 42.85%. One interesting result is `earnings22` with a much lower relative WER increase of 7.60%. We hypothesize that the smaller $\Delta_{\text{rel}}(\text{WER})$ is because the learned AMI distribution was partially relevant to `earnings22` as they are both conversational ASR datasets.

Out-of-task multilingual datasets

All measures of out-of-task forgetting relative to fine-tuning will be performed on the MLS dataset group. Details about specific data preparation can be found in section A.3. Results will be reported using the word error rate (WER) after applying the English normalizer for the English dataset and the basic normalizer for the other languages. The WERs for

the out-of-task multilingual datasets of the `tiny` model fine-tuned on AMI can be found in Table 6.4 where $\Delta_{\text{rel}}(\text{WER})$ is the relative WER reduction.

Dataset	Vanilla `tiny`	Fine-tuned `tiny`	$\Delta_{\text{rel}}(\text{WER})$ (%)
Dutch	**43.27**	66.79	-54.36
English	**12.21**	18.40	-50.70
French	**37.15**	59.93	-61.32
German	**28.31**	61.92	-118.70
Italian	**44.10**	58.60	-32.88
Polish	**38.85**	61.47	-58.22
Portuguese	**35.58**	58.52	-64.47
Spanish	**22.42**	37.81	-68.64
Non-English languages	**35.67**	57.86	-65.52
Average	**32.74**	52.93	-61.69

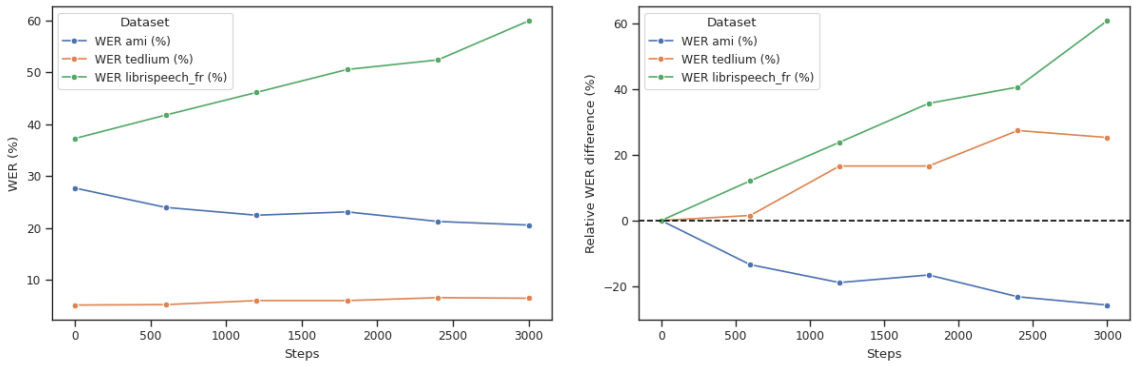
Table 6.4 WER (%) comparison between the vanilla and fine-tuned `tiny` Whisper models evaluated on the MLS dataset. Fine-tuning was performed on the train split of AMI. The best metrics per dataset are **highlighted**.

Forgetting for the out-of-task datasets (average WER increase of 65.69%) is stronger than for the out-of-distribution datasets (42.85%). Note that the drop in English was expected because the MLS English split is out-of-distribution.

Forgetting speed during fine-tuning

We saved the model every 600 steps during the fine-tuning of the `tiny` Whisper model on AMI. We took these partially trained checkpoints and evaluated them on the FAD dataset to know how fast the different WERs evolved with respect to the number of training steps. The results are shown in Figure 6.1.

As expected, the WER on the in-distribution dataset (AMI) is decreasing with respect to the number of steps. Second, we observe that the WER of the fine-tuned model on the out-of-distribution dataset (TED-LIUM) has slightly increased. Third, the WER for the out-of-task dataset (LibriSpeech French) has increased at a higher rate than for the in-distribution dataset. Thus, the further the data distribution from the training set used for fine-tuning, the faster the forgetting. We hypothesize that Whisper's implicit language model forgot about the non-English tokens while being fine-tuned on an English dataset, hence the faster drop in performance for non-English tasks.



(a) Absolute WER **(b)** Relative WER difference

Fig. 6.1 Evolution of WER (%) on the FAD dataset with respect to the training steps during the supervised fine-tuning of Whisper `tiny` on AMI. Each point corresponds to a saved and evaluated checkpoint. Relative difference is computed with respect to the vanilla model.

Implicit LM

During the process of fine-tuning Whisper on AMI, we noticed that the WER for non-English datasets exhibited a more rapid and significant increase compared to the English datasets. We hypothesize that the fine-tuned Whisper's implicit language model (LM) mostly remembers English words and has forgotten about non-English words. To confront our assumption, we will investigate the evolution of the perplexity of the implicit LM for different languages.

Background on perplexity: Perplexity is a metric used to evaluate the performance of a language model in natural language processing. It measures how well a language model can predict the probability of a sequence of words. If we have a tokenized sequence $\mathbf{X} = (x_1, \ldots, x_T)$, then the perplexity of $\mathbf{X}$ is defined as:

$$\mathrm{PPL}(\mathbf{X}) = \exp\left[-\frac{1}{T}\sum_{i=1}^{T}\log p_\theta\left(x_i \mid x_{<i}\right)\right] \tag{6.1}$$

By extension, we can define the perplexity of a given previously unseen dataset $\mathscr{D}_{\text{test}} = \{\mathbf{X}_n\}_{n\in[|1,N|]}$ where the $\mathbf{X}_n = \left(x_0^{(n)}, x_1^{(n)}, \ldots, x_t^{(n)}\right)$ by taking the average perplexity $\frac{1}{N}\sum_{n=1}^{N}\mathrm{PPL}(\mathbf{X}_n)$. The lower the perplexity value, the better the language model is at predicting tokens on this dataset. Note that a model with lower perplexity doesn't necessarily imply it will have lower WER.

Implementation: Following the procedure introduced by Gong et al. [21], we set the encoder's cross-attention weights to zero and simply fed the tokenized reference to the decoder one token at a time. The procedure is summarized in Figure 6.2.

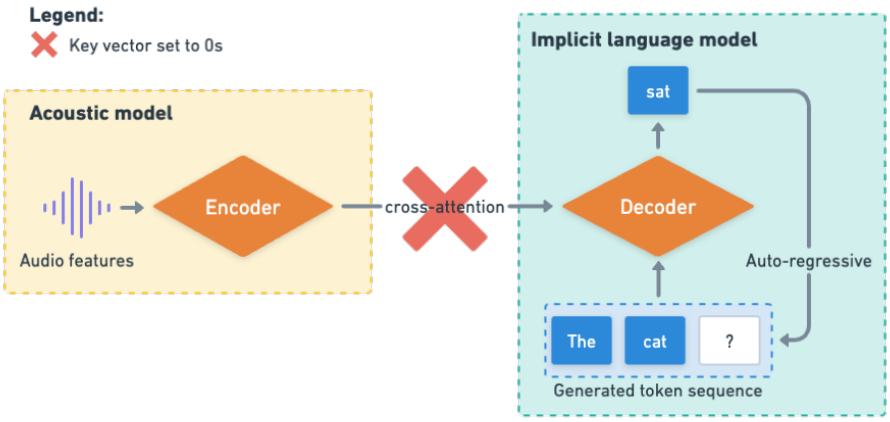


Fig. 6.2 Process to use the implicit language model in Whisper.

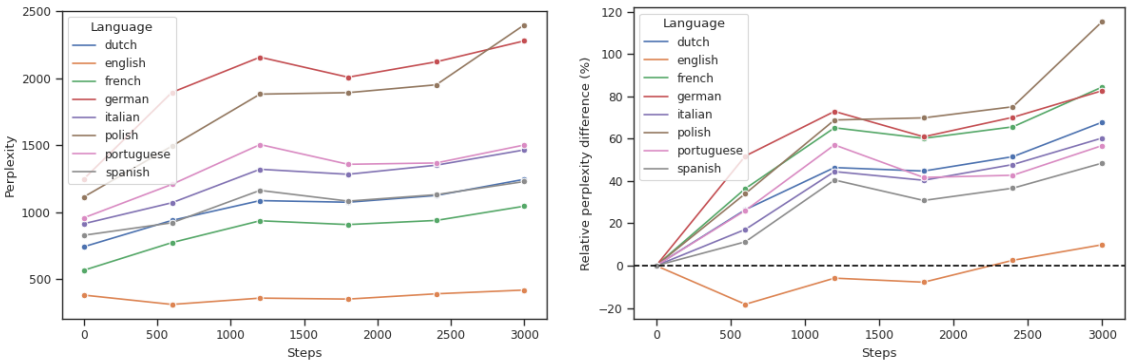
Results: We have evaluated the perplexity of Whisper's implicit LM for multilingual transcription. All non-English datasets come from the MLS dataset group (see section A.3). For the English split, we decided to compute the perplexity over all datasets from ESBDC to have a diverse enough corpus. Results are shown in Table 6.5, where $\Delta_{\text{rel}}(\text{PPL})$ is the relative perplexity reduction.

	Vanilla `tiny`	Fine-tuned `tiny`	$\Delta_{\text{rel}}(\text{PPL})$ (%)
English			
ESBDC	381.60	419.29	9.88
Non-English			
Dutch	742.98	1246.33	67.75
French	567.23	1045.76	84.36
German	1248.60	2280.35	82.63
Italian	914.80	1466.19	60.27
Polish	1115.00	2399.89	115.24
Portuguese	958.85	1502.66	56.71
Spanish	828.38	1229.63	48.44
Non-English average	910.83	1595.83	73.63

Table 6.5 Perplexity comparison between the vanilla and fine-tuned `tiny` Whisper models evaluated on the ESBDC and MLS dataset groups.

While the perplexity for English has slightly increased 9.88%, the perplexity for other languages has soared up by 73.63% on average. In terms of perplexity after fine-tuning, non-English perplexities are all more than 3 times higher than for English. Thus, this shows that the multilingual capabilities of Whisper are greatly affected by fine-tuning in a single language.

Multilingual perplexity evolution during fine-tuning: Finally, we were interested in knowing how multilingual perplexities evolved during fine-tuning. Similarly to section 52. we took the saved checkpoints obtained during fine-tuning and evaluated them using the same datasets as previously. The results are shown in Figure 6.3.



(a) Absolute perplexity **(b)** Relative perplexity difference

Fig. 6.3 Evolution of perplexity for multilingual transcription with respect to the training steps during the fine-tuning of Whisper `tiny` on AMI. Each point corresponds to a saved and evaluated checkpoint. Relative difference is computed with respect to the vanilla model.

Looking at Figure 6.3, we observe that the English perplexity is almost constant during fine-tuning while the perplexities for other languages quickly increase. It is also interesting to point out that all non-English perplexities seem to increase at the same pace.

6.2 Unsupervised knowledge distillation

While knowledge distillation (KD) is often interpolated with a supervised loss like cross-entropy (see Equation 3.4), it is also possible to distil a model in an unsupervised fashion by substituting the references with the teacher outputs. Assuming the teacher model is accurate enough, this approach would be particularly interesting for ALTA as it would cut down on the data annotation costs. Therefore, this chapter will only cover the unsupervised KD. All experiments will use `medium` as the teacher and `tiny` as the student. Finally, except if stated otherwise, the fine-tuned and distilled models were all trained on the train set of AMI until WER convergence on the validation split.

6.2.1 1-best unsupervised distillation

Teacher normalization

Following our initial discussion in subsection 5.2.2, we will investigate the impact of normalizing the teacher-generated sequences prior to KD (see subsection 5.2.2). We ran 1-best KD with different teacher normalization strategies. Results are shown in Table 6.6.

	WER (%)	del (%)	sub (%)	ins (%)
Reference (`tiny`)				
Vanilla	27.74	12.38	9.15	6.21
Default fine-tuning	20.59	5.48	10.46	4.65
Reference (`medium`)				
Vanilla	16.62	8.62	5.74	2.25
1-best KD				
(1) Default	27.30	8.86	**11.03**	7.41
(1)+(2) Remove casing/punctuation	27.07	9.12	11.27	6.68
(1)+(3) Remove trailing whitespaces	26.81	**8.91**	11.23	6.66
(1)+(2)+(3)	**25.56**	9.03	11.20	**5.32**

Table 6.6 Impact of teacher normalization during 1-best KD on the WER on the AMI test set. The best metrics for the 1-best KD results are **highlighted**.

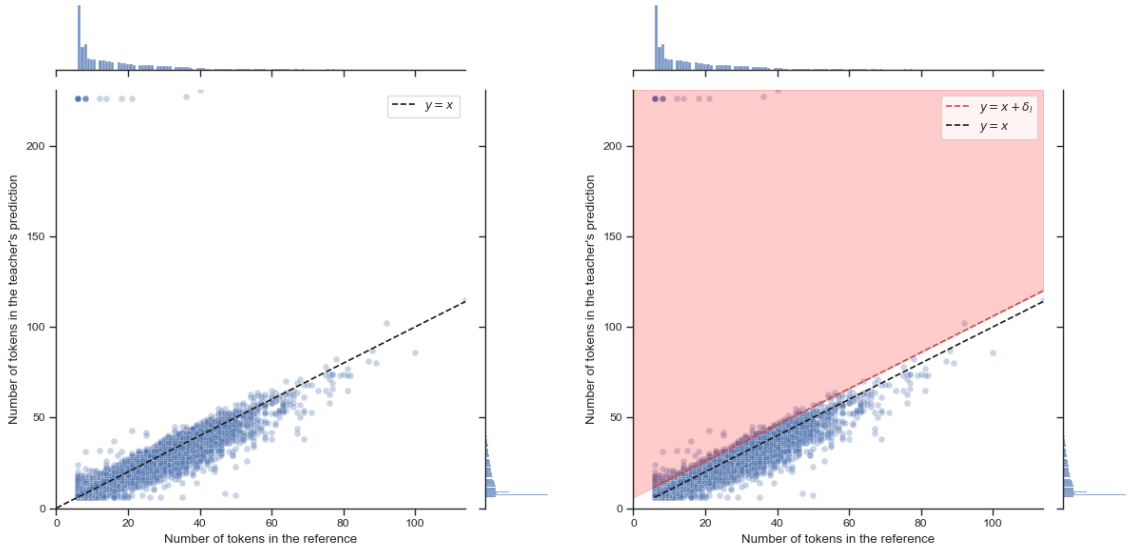
Teacher normalization was best beneficial to 1-best KD when casing, punctuation, and trailing whitespace were removed as we observed a WER drop from 27.30% to 25.56%. Consequently, we will use this teacher normalization strategy for further model distillations.

Filtering Whisper's hallucinations

As we assume that the Whisper hallucinations are detrimental to knowledge distillation, we aim to create a criterion to efficiently detect and filter out hallucinations.

Filter 1: Excess tokens

We know that hallucinations mostly consist of a large number of repeated tokens (see Figure 6.4a). Thus, we will filter out all samples where the excess number of tokens in the teacher output is strictly greater than a threshold δ_e which we will optimize using grid-search. To define a relevant grid, we will first experiment with $\delta_e = \mu_e + \sigma_e \approx 5.91$ where μ_e is the mean number of excess tokens and σ_e its standard deviation on the train split of AMI. Results are shown in Figure 6.4b and Table 6.7. Although having filtered out only less than 1% of the dataset and about 1.5% of the available audio, discarding these examples had a significant drop in WER and insertion errors for the teacher model. However, note that this filter won't be used for unsupervised distillation as it is a function of the reference text.



(a) Analysis **(b)** Proposed filter

Fig. 6.4 Impact of the excess tokens filter on the train split of AMI. The teacher model is Whisper `medium`.

% filtered rows	% filtered audio	Δ WER	Δ Ins	Δ Sub	Δ Del
0.86	1.52	0.49	0.52	0.03	-0.11

Table 6.7 Impact of the excess tokens filter on the WER on the train split of AMI.

Filter 2: Teacher gzip compression ratio

The gzip compression algorithm identifies repetitive data in a file, replaces it with shorter representations, and then encodes the compressed data using a Huffman-based algorithm. The gzip compression ratio is built on this algorithm and refers to the ratio of the compressed file size to the original file size. For example, a compression ratio of 4:1 means that the compressed file is one-fourth of the original file size. As shown in Table 6.8, the gzip compression ratio of a repetition-based hallucination sequence is indeed higher than a regular sequence as it's more compressible. Note that the gzip compression ratio criterion is preferred over a repetition count one as the repeated pattern can be more than one token long.

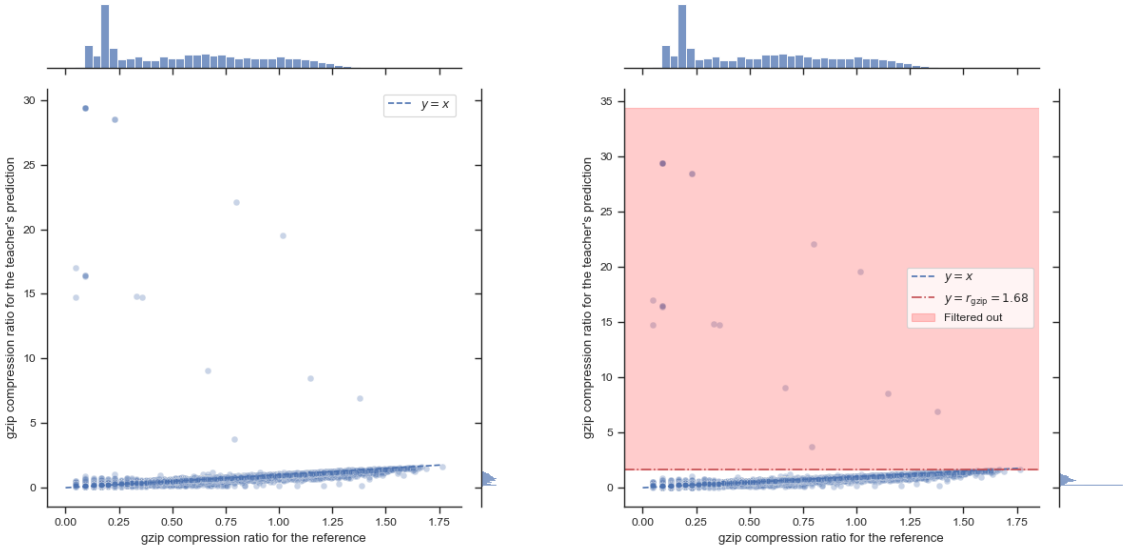
	Reference	Prediction
Text	"yeah so uh what we'll do is uh"	"So, what we will do is, we will do is, we will do is, we will do is, we will do is,"
gzip ratio	0.612	1.844

Table 6.8 Example of gzip compression ratios for Whisper `medium`'s predictions on AMI.

The distribution of the gzip ratios of the teacher predictions and of the references is shown in Figure 6.5a. While most gzip compression ratios are smaller than about 2, quite a few teachers' predictions have unusually high values of gzip ratios i.e. greater than 5. We propose to filter out samples for which the teacher predictions have a gzip ratio above a certain threshold r_{gzip}, which we will later optimize using grid-search. To determine the grid-search values, we decided to arbitrarily start with $r_{\text{gzip}} = \mu_{\text{gzip}} + 2\sigma_{\text{gzip}} \approx 1.68$ where μ_{gzip} is the mean teacher gzip ratio and σ_{gzip} its standard deviation on the train split of AMI. Results are shown in Figure 6.5b and Table 6.9. Despite having removed 0.05% of the data, we managed to obtain a decrease of 0.29% in WER and insertion rates.

% filtered rows	% filtered audio	Δ WER	Δ Ins	Δ Sub	Δ Del
0.05	0.05	0.29	0.29	0.00	0.00

Table 6.9 Impact of filtering based on the difference of the gzip ratios between the teacher and the labels on the validation split of AMI.



(a) Analysis **(b)** Proposed filter

Fig. 6.5 Impact of teacher gzip ratio filter on the train split of AMI. The teacher model is Whisper `medium`.

Filter 3: Timestamp-based filtering

Assuming Whisper can generate token-level timestamps, we hypothesize that the tokens generated when hallucinating are likely to have random timestamps or overlap. If this assumption holds, we could then use the timestamps to efficiently detect hallucinations. While not present in the original Whisper paper, OpenAI's implementation relies on the fact that cross-attention weights at each step represent the relevance of different audio parts to predicting the current token. By looking at the attention distribution when generating each token, Whisper can associate audio segments with the token being predicted at that step. So for each token in the output, Whisper records the audio segments with the highest attention. This provides token-level timestamps indicating which audio corresponds to each predicted token. Finally, dynamic time warping [63] is applied to refine the cross-attention alignments to match token boundaries better. Examples of generated timestamps are shown in Figure 6.6 and Figure 6.7.

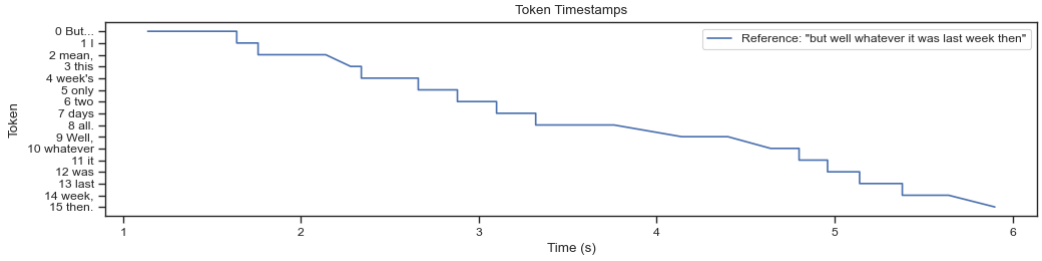


Fig. 6.6 Example of token-level time-stamping for a Whisper prediction without hallucination.

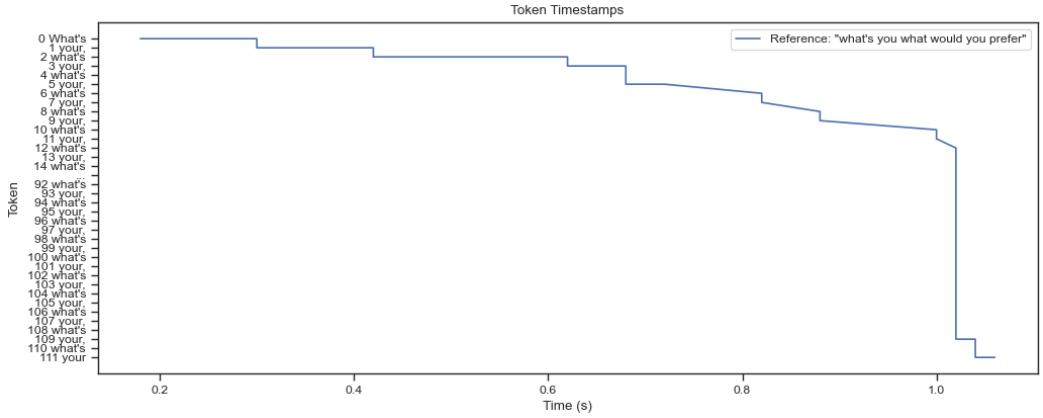
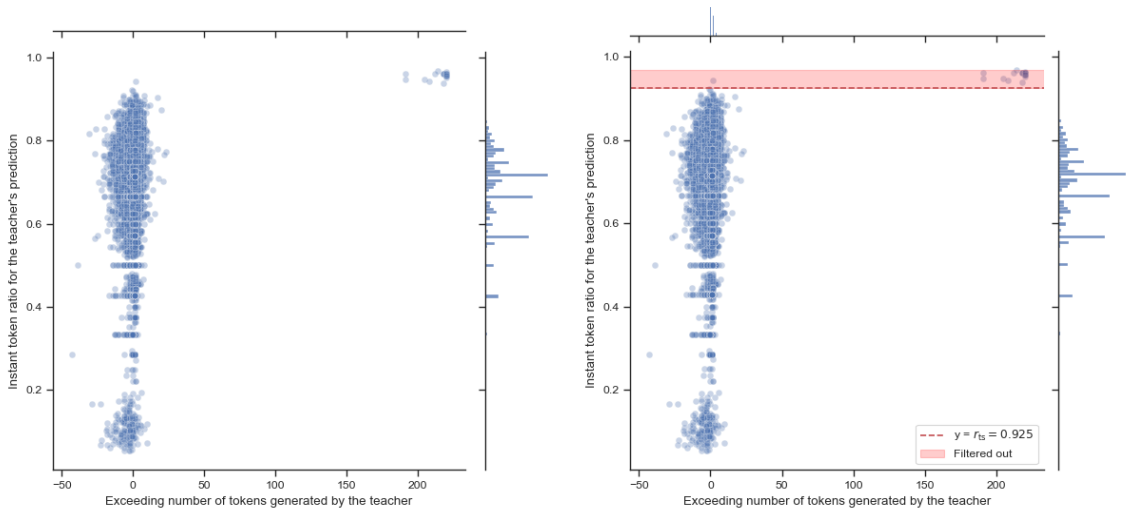


Fig. 6.7 Example of token-leve time-stamping for a Whisper prediction without hallucination.

Further analysis shows that most tokens related to hallucinations share the same timestamps. We will refer to these as *instant tokens*. Thus, can use the ratio of instant tokens in a sequence to detect and filter the hallucinations out. We can observe in Figure 6.8a that the examples with a high number of excess tokens - which contains most of the repetition-based hallucinations - all have values of instant token ratio close to 1. Thus, we will arbitrarily use $r_{\text{ts}} = 0.925$ first (see Figure 6.8b).



(a) Analysis **(b)** Proposed filter

Fig. 6.8 Evolution of the ratio of instant tokens with respect to the number of excess teacher tokens.

Recap of all filtering strategies

We ran grid-search for the three previously defined filters. To prevent data leakage, the WER evaluation was performed on the validation split of AMI. Results are shown in Table 6.10, where $\Delta_{\text{rows},(\%)}$ and $\Delta_{\text{audio},(\%)}$ are respectively the percentage of rows and of total audio that got filtered out from the training split of AMI.

The filter that provided the best results for 1-best KD is the excess tokens filter. However, we will discard it as it requires knowing the reference text, making it unsuitable for unsupervised knowledge distilltion. Instead, we decided to use the gzip ratio filter with $r_{\text{gzip}} = 1.50$ as it is the unsupervised filter that yielded the best WER. Therefore, we decided to use gzip ratio filter with $r_{\text{gzip}} = 1.50$ for the following experiments except if stated otherwise.

	Training		Validation			
	$\Delta_{\text{rows},(\%)}$	$\Delta_{\text{audio},(\%)}$	$\text{WER}_{(\%)}$	$\text{del}_{(\%)}$	$\text{sub}_{(\%)}$	$\text{ins}_{(\%)}$
Reference (`tiny`)						
Vanilla	0	0	28.47	12.38	9.15	6.21
Default fine-tuning	0	0	20.85	5.48	10.46	4.65
Reference (`medium`)	0	0				
Vanilla `medium`	0	0	17.89	9.12	5.90	2.87
1-best KD						
(1) Default	0	0	26.71	9.17	10.63	6.90
(1)+(2) Excess tokens filter						
$\delta_e = 20$	0.07	0.10	26.25	9.51	11.16	5.58
$\delta_e = 10$	0.16	0.25	25.09	8.97	11.29	4.83
$\delta_e = 5$	0.95	1.49	24.31	9.67	11.19	3.45
$\delta_e = 3$	3.22	4.71	**22.14**	**8.89**	**10.18**	**3.07**
$\delta_e = 2$	7.36	7.42	25.45	9.89	10.89	4.67
(1)+(3) gzip ratio filter						
$r_{\text{gzip}} = 1.00$	8.04	25.18	24.16	9.99	**9.90**	9.99
$r_{\text{gzip}} = 1.25$	0.73	3.18	24.08	9.28	10.11	4.69
$r_{\text{gzip}} = 1.50$	0.08	0.26	**23.87**	9.20	10.01	4.66
$r_{\text{gzip}} = 1.75$	0.05	0.06	24.28	9.67	10.00	**4.61**
$r_{\text{gzip}} = 2.00$	0.05	0.05	24.16	**8.93**	10.18	5.06
(1)+(4) Instant token ratio filter						
$r_{\text{ts}} = 0.85$	9.21	25.33	**24.16**	10.17	9.92	**4.07**
$r_{\text{ts}} = 0.875$	3.18	11.73	24.25	10.03	**9.79**	4.43
$r_{\text{ts}} = 0.90$	1.03	3.45	24.28	9.88	10.09	4.31
$r_{\text{ts}} = 0.925$	0.14	0.41	24.31	**9.14**	10.13	5.04
$r_{\text{ts}} = 0.95$	0.05	0.05	24.51	9.50	**9.92**	5.09

Table 6.10 Evolution of the WER on the validation split of AMI. The best metric for each filter is **highlighted**.

Results of 1-best unsupervised KD

With our new filtering strategy, we ran 1-best unsupervised KD using teacher normalization and the gzip compression ratio. Results are shown in Table 6.11.

	WER (%)	del (%)	sub (%)	ins (%)
Reference (`tiny`)				
Vanilla	27.74	12.38	9.15	6.21
Default fine-tuning	20.59	5.48	10.46	4.65
Reference (`medium`)				
Vanilla	16.62	8.62	5.74	2.25
1-best unsupervised KD				
(1) Default	27.30	**8.86**	11.03	7.41
(1)+(2) Teacher normalization	25.56	9.03	11.20	5.32
(1)+(3) gzip ratio filter	**22.80**	9.08	**10.39**	**3.33**
(1)+(2)+(3)	24.69	8.97	11.31	4.41

Table 6.11 WER metrics for 1-best unsupervised KD compared to other Whisper models on AMI test. The best metrics for the distilled model are **highlighted**.

First, 1-best unsupervised KD achieves the best performance with the gzip compression ratio filter. It is interesting to notice that while this filter only removed 0.08% of the examples and 0.26% of the audio from the training set (see Table 6.10, it decreased the WER from 27.30% to 22.80% i.e. a relative drop of 24.58%. On a different note, combining teacher post-processing and gzip compression filtering has slightly worsened the results compared to using the gzip filter alone. Moreover, the high deletion rates of the vanilla teacher `medium` and the distilled student are peculiar as they are about two times larger compared to default fine-tuning. We will further investigate this last point.

Analysis

To begin with, we would like to confirm that the student has correctly learned from the teacher. If that is the case, the orthographic WER (i.e. WER without prior normalization, denoted $\text{WER}_{\perp}$) of the student against the teacher's predictions should be similar to the orthographic WER of the fine-tuned model against the reference as both the teacher's predictions and the reference play the same role. Results are shown in Table 6.12.

We observe little to no difference between the two set of WER scores. Thus, we believe the student has mimicked the teacher to the best of its capabilities. Therefore, we decided to compare the student predictions with reference and observed a high amount of disfluency

	$\text{WER}_{\perp}^{(\%)}$	$\text{del}_{\perp}^{(\%)}$	$\text{sub}_{\perp}^{(\%)}$	$\text{ins}_{\perp}^{(\%)}$
Fine-tuned model against the reference	20.91	4.67	12.03	4.21
1-best student against the teacher's predictions	20.10	4.15	10.98	4.97

Table 6.12 Orthographic WER metrics comparison for default fine-tuning and 1-best KD.

removal similar to what was observed in section 5.4.2. Hence, it seems difficult to match the fine-tuning deletion rate because the teacher model is not aware of the formatting used when annotating the AMI corpus. We hypothesize that fine-tuning or soft-prompting [46] the teacher on a small amount of annotated AMI data prior to KD should fix the issue.

6.2.2 Word-level unsupervised distillation

To make the word-level KD unsupervised, we replaced the reference with the teacher's prediction. Note that this method is all the more interesting as, contrarily to the supervised word-level KD, the cross-entropy loss and the KL divergence are based on the same underlying teacher distribution.

Results

To begin with, we performed a grid search to optimize the hyperparameters for word-level KD. For simplicity, we only tried different values of α and kept $\tau = 1$ for the temperature based on empirical results from the literature [45, 65]. Capitalizing the results from subsection 6.2.1, we will also apply the gzip compression ratio filter. Results for AMI validation are shown in Table 6.13.

	$\text{WER}_{(\%)}$	$\text{del}_{(\%)}$	$\text{sub}_{(\%)}$	$\text{ins}_{(\%)}$
Reference (`tiny`)				
Vanilla	28.47	12.38	9.15	6.21
Default fine-tuning	20.85	10.46	5.48	4.65
Reference (`medium`)				
Vanilla	17.89	9.12	5.90	2.87
1-best unsupervised KD				
With gzip filter	**23.87**	**9.20**	**10.01**	4.66
Word-level unsupervised KD				
$\alpha = 0.3,\ \tau = 1$, gzip filter	25.56	10.00	10.07	5.48
$\alpha = 0.5,\ \tau = 1$, gzip filter	25.53	9.97	10.08	5.48
$\alpha = 0.8,\ \tau = 1$, gzip filter	25.07	9.89	10.08	5.10
$\alpha = 0.9,\ \tau = 1$, gzip filter	24.66	9.75	10.08	4.82
$\alpha = 0.95,\ \tau = 1$, gzip filter	24.32	9.69	10.03	**4.60**

Table 6.13 WER metrics for word-level unsupervised KD compared to other Whisper models on AMI validation. The best metrics for the distilled model are **highlighted**.

Analysis

We observe that the higher α, the lower the WER. As expected, we get closer to the 1-best unsupervised results when $\alpha \to 1$. Thus, the KL divergence seems to be detrimental to distillation.

6.2.3 K-best unsupervised distillation

Since 1-best KD gave better results than word-level KD, we will try to improve on the resulting WER by investigating the K-best unsupervised KD. We used the same gzip filter as for 1-best but this time we filtered out every example for which at least one of the K sequences generated by beam-search wouldn't match the gzip criterion.

Results

First, we evaluated the different 3-best strategies for the AMI validation split. $K = 3$ was arbitrarily chosen for all the following experiments. Results are shown in Table 6.14.

	$\text{WER}_{(\%)}$	$\text{del}_{(\%)}$	$\text{sub}_{(\%)}$	$\text{ins}_{(\%)}$
Reference (`tiny`)				
Vanilla (greedy search)	28.47	12.38	9.15	6.21
Vanilla (3-beam search)	31.42	8.69	11.12	11.61
Default fine-tuning	20.85	10.46	5.48	4.65
Reference (`medium`)				
Vanilla	17.89	9.12	5.90	2.87
1-best unsupervised KD				
With gzip filter	**23.87**	**9.20**	**10.01**	4.66
Word-level unsupervised KD				
$\alpha = 0.95,\ \tau = 1$, gzip filter	24.32	9.69	10.03	4.60
3-best unsupervised KD				
Uniform + gzip filter	24.83	10.14	10.68	4.01
Ranked ($\beta = 1$) + gzip filter	25.06	10.48	10.43	4.15
Ranked ($\beta = 2$) + gzip filter	25.08	10.56	10.47	4.05
Ranked ($\beta = 5$) + gzip filter	25.11	10.59	10.47	4.05
Ranked ($\beta = 10$) + gzip filter	24.96	10.60	10.47	**3.90**

Table 6.14 WER metrics for K-best unsupervised KD compared to other Whisper models on AMI validation. The best metrics for the distilled model are **highlighted**.

The best 3-best method is the uniform KD. Thus, we will keep this method for the final WER evaluation on the test split of AMI. Note that even the ranked K-best with $\beta = 10$, which is equivalent to only considering the best sequence for the 3-beam, is more than 1 WER point behind 1-best KD. We can explain this gap because the WER for 3-beam on AMI is actually higher than for the 1-best i.e. greedy search in our study.

	WER (%)	del (%)	sub (%)	ins (%)
Reference (`tiny`)				
Vanilla	27.74	12.38	9.15	6.21
Default fine-tuning	20.59	5.48	10.46	4.65
Reference (`medium`)				
Vanilla	16.62	8.62	5.74	2.25
1-best KD				
With gzip filter	**22.80**	9.08	**10.39**	3.33
Word-level unsupervised KD				
$\alpha = 0.95,\ \tau = 1$, gzip filter	23.93	9.39	10.64	3.89
3-best unsupervised KD				
Uniform + gzip filter	23.01	**8.99**	11.07	**2.95**

Table 6.15 WER metrics for the 3-best unsupervised KD compared to other Whisper models on AMI test. The best metrics for the distilled model are **highlighted**.

Analysis

3-best didn't manage to beat 1-best KD. Again we believe that this is due to the poor performance of the 3-beam search on AMI. We hypothesize that having a dataset for which the WER would decrease with the K beam size would be beneficial to K-best.

Wrap-up The performance of all the distillation methods is summarized in Figure 6.9. The best results for unsupervised distillation on AMI were obtained using the 1-best KD with the gzip filter ($r_{\text{gzip}} = 1.50$), decreasing the WER on the test split from 27.74% to 22.80% i.e. an absolute drop of 4.94% (or a relative drop of 17.81%). Interestingly, we achieved a 95% size reduction from `medium` to `tiny` for a relative WER increase of 67%. While we still haven't managed to beat the result of supervised fine-tuning (20.59%), we hypothesize that fine-tuning or soft-prompting the teacher on a small subset of AMI would fix the formatting discrepancy mentioned in section 62 and close the WER gap.

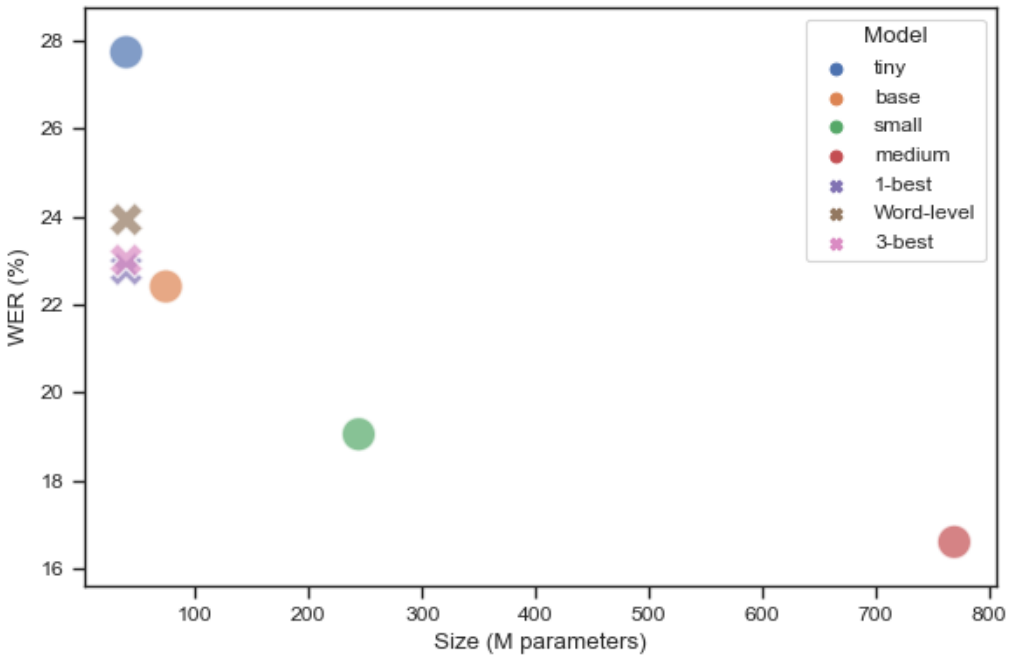


Fig. 6.9 Example of time-stamping for a Whisper prediction without hallucination.

6.3 Continual learning

6.3.1 Elastic Weight Consolidation

EWC requires estimating the EWC parameters and the observed Fisher information matrix of the pre-trained model. While estimating the weights is straightforward, estimating the Fisher information matrix demands running inference on the dataset the model was trained on previously. In the case of Whisper, we don't have access to the original dataset used by OpenAI during pre-training. Thus, we will use our target dataset (AMI) as a substitute for estimating the EWC parameters. We hypothesize that even with this poor dataset proxy, the fine-tuned model can preserve a fraction of its original multi-task capabilities. All models will be evaluated on the FAD dataset group (see section A.4).

First, we need to choose a value for the strength of the EWC regularization λ. Thus, we will perform a grid search for $\lambda \in \{\texttt{1e-4}, \texttt{1e-3}, \texttt{1e-2}, \texttt{1e-1}, \texttt{1e-0}\}$. Results are shown in Table 6.16.

Name	AMI	MLS French	TED-LIUM
References			
Vanilla	27.74	37.27	5.18
Default fine-tuning ($\lambda = 0$)	**20.85**	59.21	6.49
Fine-tuning with EWC			
$\lambda = \texttt{1e-4}$	22.51	48.87	6.27
$\lambda = \texttt{1e-3}$	25.87	41.13	6.09
$\lambda = \texttt{1e-2}$	26.47	40.41	5.80
$\lambda = \texttt{1e-1}$	27.67	39.62	5.51
$\lambda = \texttt{1e+0}$	27.15	**38.55**	**5.24**

Table 6.16 Impact of λ on the WER (%) for the FAD dataset group after fine-tuning on AMI for 3000 steps with EWC. The best metrics among the fine-tuned models are **highlighted**.

As expected, the higher the value of λ, the less important the forgetting on the out-of-task distribution (MLS French). We are nonetheless concerned that EWC has only slowed down fine-tuning. Thus, we compared the EWC results with the different checkpoints obtained during the default fine-tuning from section 52 for AMI and MLS FR. The results are shown in Figure 6.10. We observed that fine-tuning with EWC performs very similarly to default fine-tuning. To further confirm our assumption, we fitted a linear regression which had an R^2 score of 0.954. Thus we can assume that the only impact of EWC was to slow down fine-tuning.

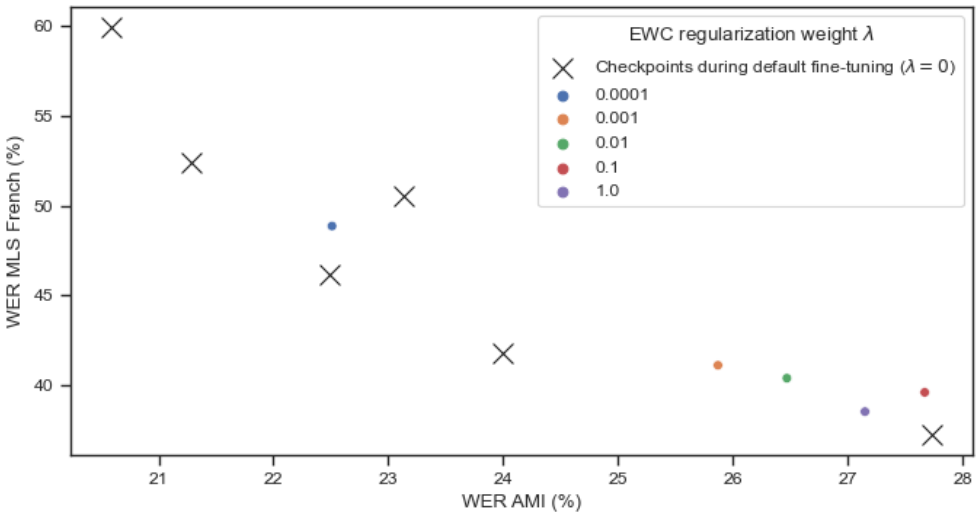


Fig. 6.10 WER pairplot on AMI and MLS French test splits with respect to λ. The points for default fine-tuning correspond to the different checkpoints every 600 steps.

We will now assume that we have an annotated out-of-task dataset at hand that can be used to estimate the Fisher information matrix. In this view, we will arbitrarily use the MLS FR training split. After performing a grid-search for $\lambda \in \{\texttt{1e-4}, \texttt{1e-3}, \texttt{1e-2}, \texttt{1e-1}, \texttt{1e-0}\}$, we decided to use $\lambda = \texttt{1e-4}$. Results are shown in Table 6.17. Note that we also experimented with averaging the EWC parameters obtained with AMI and MLS FR (see the AMI + MLS FR row).

	AMI	TED-LIUM	MLS FR
References			
Vanilla	27.74	5.16	37.15
Default fine-tuning	**20.59**	6.49	59.93
EWC with parameters estimated from			
AMI	22.51	**6.27**	48.87
MLS FR	21.80	7.27	**38.06**
AMI + MLS FR	21.38	7.32	38.49

Table 6.17 Impact of the dataset used for estimating the EWC parameters on the WER (%) for the FAD dataset group. Fine-tuning with tEWC was performed on AMI for 3000 steps. The best metrics among the fine-tuned models are **highlighted**.

Looking at Table 6.17, using fine-tuning with EWC had the expected behavior for tackling forgetting: we lowered the WER on the target dataset (AMI) and we preserved the original Whisper performance on the out-of-task distribution (MLS FR). It is interesting to notice that EWC worsened the performance of the tasks that were not covered by the dataset used for estimating the EWC parameters. Results for other languages are shown in Table 6.18, where $\Delta_{\text{rel}}^{(EWC)}(\text{WER})$ is the relative difference between the WER of the fine-tuned model without EWC and with EWC.

Dataset	Vanilla	Fine-tuned	Fine-tuned with EWC	$\Delta_{\text{rel}}^{(EWC)}(\text{WER})$ (%)
Dutch	43.27	66.79	**52.61**	21.23
English	12.21	**18.40**	19.09	-3.75
French	37.15	59.93	**38.06**	36.49
German	28.31	61.92	**35.79**	42.20
Italian	44.10	58.60	**49.83**	14.97
Polish	38.85	61.47	**53.45**	13.05
Portuguese	35.58	58.52	**45.07**	22.98
Spanish	22.42	37.81	**28.42**	24.83
Non-English (avg)	35.67	57.86	43.32	25.13
Average	32.74	52.93	40.29	23.88

Table 6.18 Impact of EWC on the out-of-task WER of Whisper `tiny` on the MLS dataset. Fine-tuning was performed on the train split of AMI. The best metrics among the fine-tuned models and per dataset are **highlighted**.

All non-English languages benefited from EWC regularization. This is all the more surprising as the EWC weights were estimated using only French data. We hypothesize that a fraction of the important weights for transcribing French is shared with the ones used for transcribing other languages. Therefore, we hypothesize that the more similar the language to French, the more important weights they have in common.

We are also interested in knowing the evolution of WER for both default and EWC using the parameters estimated from MLS FR. Results are shown in Figure 6.11.

Looking at Figure 6.11, we observe that the WER evolution of AMI with EWC is very similar to default fine-tuning. Moreover, EWC managed to keep the WER for MLS FR under no more than 4% of the vanilla performance for the training on 77h of audio. Nonetheless, preserving French is detrimental to preserving the out-of-distribution performance on TED-LIUM as the relative WER difference rapidly jumped to more than 30%. For comparison, the same WER increase was much slower for default fine-tuning.

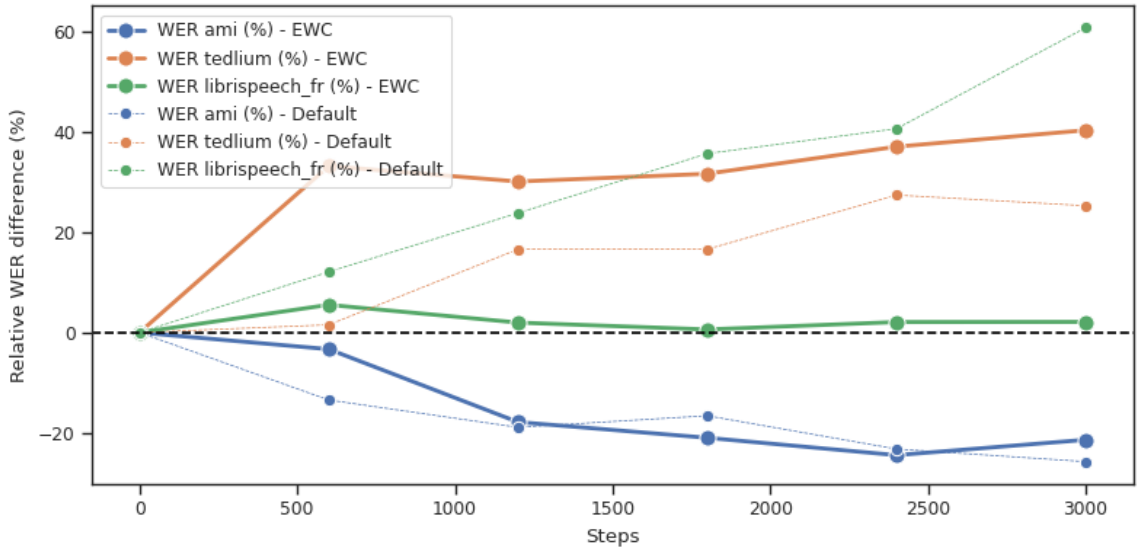


Fig. 6.11 Evolution of WER (%) on the FAD dataset with respect to the training steps during the fine-tuning of Whisper `tiny` on AMI. Each point corresponds to a saved and evaluated checkpoint. Relative difference is computed with respect to the vanilla model.

6.3.2 Task Alignment Consolidation

In the previous section, we used EWC to apply regularization within the parameter space. However, we hypothesize that the closer to the generation we regularize, the more likely we are to preserve the previous tasks learned by Whisper. For this reason, we are interested in regularizing directly from the prediction space. As explained in subsection 4.2.2, TAC is interesting for such a scheme as it doesn't need the original dataset. Thus, we will assume that the target set (AMI) is the only dataset available.

The first step in TAC is to prompt Whisper to generate pseudo-transcription i.e. to transcribe French words from English audio. As expected, Whisper generates French text that sounds similar to the English words from the audio (see the first row of Table 6.19).

References	**Transcribe English speech as French (*pseudo-French*)**	**gzip compression ratio**
nor is mister quilter's manner less interesting than his matter	N'orise Mr. Quilterre s'en m'en est intéressant thané le mâtre.	0.79
he tells us that at this festive season of the year with christmas and roast beef looming before us similes drawn from eating and its results occur most readily to the mind	Il a dit que cette fois-tu vu une fois-tu vu une fois-tu vu une fois-tu vu une fois-tu vu une [...] fois-tu vu une fois	12.86

Table 6.19 Examples of pseudo-transcriptions generated by the vanilla Whisper `tiny` on arbitrary examples from LibriSpeech clean. The gzip compression ratios are for the pseudo-French transcriptions.

However, we can observe in Figure 6.12 that Whisper frequently suffered from repetition-based hallucinations like the one shown in the last row of Table 6.19. We hypothesize that this is due to the poor performance of the vanilla `tiny` model (37.27% i.e. more than 1 word out of 3 is wrong). This raises some concerns about the efficiency of the TAC method on Whisper `tiny`.

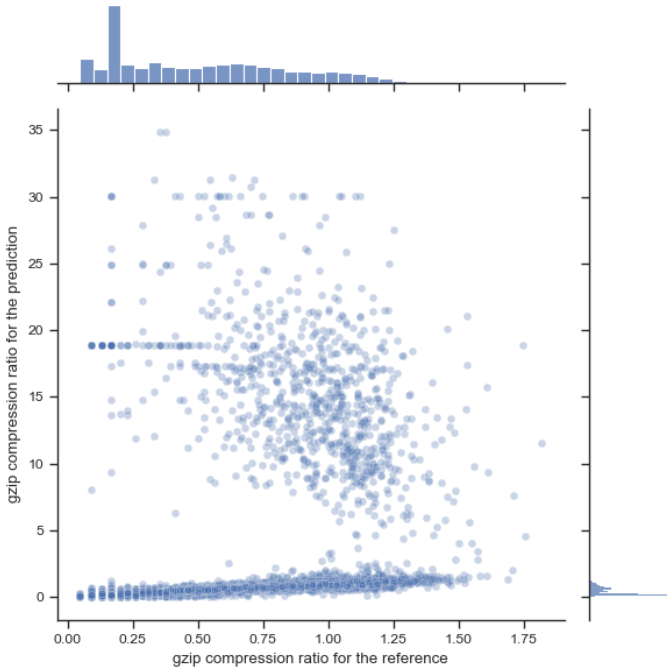


Fig. 6.12 Distribution of the gzip compression ratio of the references and the predictions of `tiny` on the AMI test split.

We performed a grid search to optimize the strength of the TAC regularization γ. Results are shown in Table 6.20.

Name	WER AMI (%)	WER MLS French (%)
References		
Vanilla	28.47	37.15
Default fine-tuning ($\gamma = 0$)	23	45.13
Fine-tuning with TAC		
$\gamma = 0.01$	21.99	49.38
$\gamma = 0.1$	23.95	48.29
$\gamma = 1$	26.84	48.99

Table 6.20 Impact of γ on the WER (%) of Whisper `tiny` fine-tuned using TAC and evaluated on the validation split of AMI.

Looking at Table 6.20, TAC is demonstrated to be an ineffective way of preventing forgetting during fine-tuning as the out-of-task performances on MLS WER are even worse than the one obtained with default fine-tuning. The poor results can be explained by two reasons. First, TAC assumes that the Whisper's nodes for transcribing different languages are entangled. However, we know from subsection 6.3.1 that EWC, which is built on the opposite assumption, performs quite well on `tiny`. Thus, we expected such results for TAC

on the same model. Our second assumption is that TAC performs poorly because of the poor French transcription capabilities of `tiny` (37.15%) on the MLS FR test set (i.e. more than 1 word out of 3 is wrong). In preparation for future work, we analyzed pseudo-transcriptions for larger sizes of Whisper. To our surprise, not only did the pseudo-transcriptions sound more like the original English sentence, but the large models (`small` and `medium`) seem to have translated the English audio into French. Hence, we hypothesize TAC would be more effective on the larger Whisper models.

Reference	mister quilter is the apostle of the middle classes and we are glad to welcome his gospel
French translation	monsieur quilter est l'apôtre des classes moyennes et nous sommes heureux d'accueillir son gospel
`tiny`	Le plus de la chasse est de la classe et nous débrouillons l'air de l'air de la chasse.
`base`	Mr. Quilter est le passé de la classe de la ville, et nous sommes glad de bienvenue son gosse-boule.
`small`	Mr Quilter est l'époche de la classe moyenne et nous sommes heureux d'accueillir son gospel.
`medium`	Mr. Quilter est l'apostle des classes du milieu et nous sommes heureux de la bienvenue dans son Gospèle.

Table 6.21 Example of pseudo-French transcriptions for the different Whisper sizes.

Chapter 7

Conclusion

The experiments performed in this report first show that naively using the raw teacher outputs for distillation is inefficient. When used with Whisper `tiny` as the student and Whisper `medium` as the teacher, 1-best distillation yields a poor WER improvement from 27.74% to 27.30% on the AMI test set. An investigation was conducted to improve distillation performance by normalizing the teacher's predictions and filtering out hallucinations. This approach is sensible for two reasons: firstly, the pre-trained Whisper model might not have the same formatting as the target dataset, and secondly, the few hallucinations generated by Whisper are demonstrated to deteriorate the student's performance during distillation significantly. Notably, filtering out the teacher's transcriptions with high values of gzip compression ratio is demonstrated to be quite effective: while it removes only 0.08% of the examples and 0.26% of the audio from the training set, the subsequent 1-best yields a much more reasonable WER decrease from from 27.74% to 22.80% on AMI test, i.e. a 17.81% relative improvement. If further research is done into distillation, it might be interesting to study the relationship of the model compression ratio with expected performance gains. With this in mind, we hypothesize that distilling `medium` into `tiny` for a size reduction of 95% may have been too ambitious as we observed a relative WER increase of 67%. While our distillation methods are all based on only the output of the teacher model (*response-based knowledge*), a complementary experiment could involve the student mimicking the intermediate layers of the teacher [59, 65] (*feature-based knowledge*) to distil in a representation learning fashion [8]. Additionally, using soft-prompting on a small subset of the target dataset to adapt the teacher's formatting seems promising. Further work might also involve trying different decoding strategies to reduce the frequency of hallucinations (e.g., decoding using constrained beam search [30], the N-gram penalty [54], or *Locally Typical Sampling* [49]).

For continual learning, estimating the Fisher information matrix using a non-English transcription dataset and fine-tuning it with the EWC regularization proves to be an im-

pressive candidate for continual learning. Not only does it manage to keep the WER for French transcription under 4% of the vanilla performance for the whole training, but it also significantly reduces forgetting for other non-English transcription tasks: compared to default fine-tuning, EWC achieves an average relative WER drop of 25.13% on the non-English datasets from Multilingual LibriSpeech. Further work on this task may be best focused on trying and combining different datasets to estimate the EWC parameters. On the other hand, continual learning with limited access to the original dataset proved to be an arduous task as TAC is demonstrated to be inefficient in preserving the multilingual capabilities of Whisper during fine-tuning on an English dataset. Moreover, using the KL divergence to compare the pseudo-transcriptions may also lead to better performance for TAC. Finally, it might be interesting to focus on larger Whisper models as new candidates for TAC: we believe that having highly entangled cross-lingual representations might facilitate preserving multilingual transcriptions using only English.

References

[1] Aich, A. (2021). Elastic Weight Consolidation (EWC): Nuts and Bolts.

[2] Aljundi, R., Babiloni, F., Elhoseiny, M., Rohrbach, M., and Tuytelaars, T. (2018). Memory Aware Synapses: Learning what (not) to forget.

[3] Ardila, R., Branson, M., Davis, K., Henretty, M., Kohler, M., Meyer, J., Morais, R., Saunders, L., Tyers, F. M., and Weber, G. (2020). Common Voice: A Massively-Multilingual Speech Corpus.

[4] Babu, A., Wang, C., Tjandra, A., Lakhotia, K., Xu, Q., Goyal, N., Singh, K., von Platen, P., Saraf, Y., Pino, J., Baevski, A., Conneau, A., and Auli, M. (2021). XLS-R: Self-supervised Cross-lingual Speech Representation Learning at Scale.

[5] Baevski, A., Zhou, H., Mohamed, A., and Auli, M. (2020). Wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations.

[6] Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural Machine Translation by Jointly Learning to Align and Translate.

[7] Bengio, S., Vinyals, O., Jaitly, N., and Shazeer, N. (2015). Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks.

[8] Bengio, Y., Courville, A., and Vincent, P. (2012). Representation Learning: A Review and New Perspectives.

[9] Carletta, J., Ashby, S., Bourban, S., Flynn, M., Guillemot, M., Hain, T., Kadlec, J., Karaiskos, V., Kraaij, W., Kronenthal, M., Lathoud, G., Lincoln, M., Lisowska, A., McCowan, I., Post, W., Reidsma, D., and Wellner, P. (2006). The AMI Meeting Corpus: A Pre-announcement. In Renals, S. and Bengio, S., editors, *Machine Learning for Multimodal Interaction*, volume 3869, pages 28–39. Springer Berlin Heidelberg, Berlin, Heidelberg.

[10] Castro, F. M., Marín-Jiménez, M. J., Guil, N., Schmid, C., and Alahari, K. (2018). End-to-End Incremental Learning.

[11] Chen, G., Chai, S., Wang, G., Du, J., Zhang, W.-Q., Weng, C., Su, D., Povey, D., Trmal, J., Zhang, J., Jin, M., Khudanpur, S., Watanabe, S., Zhao, S., Zou, W., Li, X., Yao, X., Wang, Y., Wang, Y., You, Z., and Yan, Z. (2021a). GigaSpeech: An Evolving, Multi-domain ASR Corpus with 10,000 Hours of Transcribed Audio.

[12] Chen, P. H., Si, S., Li, Y., Chelba, C., and Hsieh, C.-j. (2018). GroupReduce: Block-Wise Low-Rank Approximation for Neural Language Model Shrinking.

[13] Chen, S., Wang, C., Chen, Z., Wu, Y., Liu, S., Chen, Z., Li, J., Kanda, N., Yoshioka, T., Xiao, X., Wu, J., Zhou, L., Ren, S., Qian, Y., Qian, Y., Wu, J., Zeng, M., Yu, X., and Wei, F. (2021b). WavLM: Large-Scale Self-Supervised Pre-Training for Full Stack Speech Processing.

[14] Chung, Y.-A., Zhang, Y., Han, W., Chiu, C.-C., Qin, J., Pang, R., and Wu, Y. (2021). W2v-BERT: Combining Contrastive Learning and Masked Language Modeling for Self-Supervised Speech Pre-Training.

[15] Courbariaux, M., Bengio, Y., and David, J.-P. (2015). BinaryConnect: Training Deep Neural Networks with binary weights during propagations.

[16] Dao, T., Fu, D. Y., Ermon, S., Rudra, A., and Ré, C. (2022). FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness.

[17] Del Rio, M., Delworth, N., Westerman, R., Huang, M., Bhandari, N., Palakapilly, J., McNamara, Q., Dong, J., Zelasko, P., and Jette, M. (2021). Earnings-21: A Practical Benchmark for ASR in the Wild. In *Interspeech 2021*, pages 3465–3469.

[18] Fan, A., Lewis, M., and Dauphin, Y. (2018). Hierarchical Neural Story Generation.

[19] French, R. (1999). Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128–135.

[20] Gandhi, S., von Platen, P., and Rush, A. M. (2022). ESB: A Benchmark For Multi-Domain End-to-End Speech Recognition.

[21] Gong, Z., Saito, D., Li, S., Kawai, H., and Minematsu, N. (2022). Can We Train a Language Model Inside an End-to-End ASR Model? - Investigating Effective Implicit Language Modeling.

[22] Graves, A. (2012). Sequence Transduction with Recurrent Neural Networks.

[23] Graves, A., Fernández, S., Gomez, F., and Schmidhuber, J. (2006). Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd International Conference on Machine Learning - ICML '06*, pages 369–376, Pittsburgh, Pennsylvania. ACM Press.

[24] Gulati, A., Qin, J., Chiu, C.-C., Parmar, N., Zhang, Y., Yu, J., Han, W., Wang, S., Zhang, Z., Wu, Y., and Pang, R. (2020). Conformer: Convolution-augmented Transformer for Speech Recognition.

[25] Han, S., Pool, J., Tran, J., and Dally, W. J. (2015). Learning both Weights and Connections for Efficient Neural Networks.

[26] Hassibi, B. and Stork, D. (1992). Second order derivatives for network pruning: Optimal Brain Surgeon. In Hanson, S., Cowan, J., and Giles, C., editors, *Advances in Neural Information Processing Systems*, volume 5. Morgan-Kaufmann.

[27] He, Y., Sainath, T. N., Prabhavalkar, R., McGraw, I., Alvarez, R., Zhao, D., Rybach, D., Kannan, A., Wu, Y., Pang, R., Liang, Q., Bhatia, D., Shangguan, Y., Li, B., Pundak, G., Sim, K. C., Bagby, T., Chang, S.-y., Rao, K., and Gruenstein, A. (2018). Streaming End-to-end Speech Recognition For Mobile Devices.

[28] Hernandez, F., Nguyen, V., Ghannay, S., Tomashenko, N., and Estève, Y. (2018). TED-LIUM 3: Twice as much data and corpus repartition for experiments on speaker adaptation. volume 11096, pages 198–208.

[29] Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the Knowledge in a Neural Network.

[30] Hokamp, C. and Liu, Q. (2017). Lexically Constrained Decoding for Sequence Generation Using Grid Beam Search. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1535–1546, Vancouver, Canada. Association for Computational Linguistics.

[31] Holtzman, A., Buys, J., Du, L., Forbes, M., and Choi, Y. (2020). The Curious Case of Neural Text Degeneration.

[32] Hsu, W.-N., Bolte, B., Tsai, Y.-H. H., Lakhotia, K., Salakhutdinov, R., and Mohamed, A. (2021). HuBERT: Self-Supervised Speech Representation Learning by Masked Prediction of Hidden Units.

[33] Huszár, F. (2017). On Quadratic Penalties in Elastic Weight Consolidation.

[34] Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., and Kalenichenko, D. (2017). Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference.

[35] Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. (2020). Scaling Laws for Neural Language Models.

[36] Kim, J. and Kang, P. (2021). K-Wav2vec 2.0: Automatic Speech Recognition based on Joint Decoding of Graphemes and Syllables.

[37] Kim, J.-H. and Woodland, P. C. (2003). A combined punctuation generation and speech recognition system and its performance enhancement using prosody. *Speech Communication*, 41(4):563–577.

[38] Kim, Y. and Rush, A. M. (2016). Sequence-Level Knowledge Distillation.

[39] Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., and Hadsell, R. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526.

[40] Lhoest, Q., del Moral, A. V., Jernite, Y., Thakur, A., von Platen, P., Patil, S., Chaumond, J., Drame, M., Plu, J., Tunstall, L., Davison, J., Šaško, M., Chhablani, G., Malik, B., Brandeis, S., Scao, T. L., Sanh, V., Xu, C., Patry, N., McMillan-Major, A., Schmid, P., Gugger, S., Delangue, C., Matussière, T., Debut, L., Bekman, S., Cistac, P., Goehringer, T., Mustar, V., Lagunas, F., Rush, A. M., and Wolf, T. (2021). Datasets: A Community Library for Natural Language Processing.

[41] Li, X., Qin, T., Yang, J., and Liu, T.-Y. (2016). LightRNN: Memory and Computation-Efficient Recurrent Neural Networks.

[42] Li, Z. and Hoiem, D. (2017). Learning without Forgetting.

[43] Liu, A. T., Yang, S.-w., Chi, P.-H., Hsu, P.-c., and Lee, H.-y. (2019). Mockingjay: Unsupervised Speech Representation Learning with Deep Bidirectional Transformer Encoders.

[44] Loshchilov, I. and Hutter, F. (2019). Decoupled Weight Decay Regularization.

[45] Ma, R., Liu, Q., and Yu, K. (2019). Highly Efficient Neural Network Language Model Compression Using Soft Binarization Training. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 62–69, SG, Singapore. IEEE.

[46] Ma, R., Qian, M., Gales, M. J. F., and Knill, K. M. (2023). Adapting an ASR Foundation Model for Spoken Language Assessment.

[47] MacKay, D. J. C. (1992). A Practical Bayesian Framework for Backpropagation Networks. *Neural Computation*, 4(3):448–472.

[48] McCloskey, M. and Cohen, N. J. (1989). Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem. In *Psychology of Learning and Motivation*, volume 24, pages 109–165. Elsevier.

[49] Meister, C., Pimentel, T., Wiher, G., and Cotterell, R. (2022). Locally Typical Sampling.

[50] Michel, P., Levy, O., and Neubig, G. (2019). Are Sixteen Heads Really Better than One?

[51] Micikevicius, P., Narang, S., Alben, J., Diamos, G., Elsen, E., Garcia, D., Ginsburg, B., Houston, M., Kuchaiev, O., Venkatesh, G., and Wu, H. (2018). Mixed Precision Training.

[52] O'Neill, P. K., Lavrukhin, V., Majumdar, S., Noroozi, V., Zhang, Y., Kuchaiev, O., Balam, J., Dovzhenko, Y., Freyberg, K., Shulman, M. D., Ginsburg, B., Watanabe, S., and Kucsko, G. (2021). SPGISpeech: 5,000 hours of transcribed financial audio for fully formatted end-to-end speech recognition.

[53] Panayotov, V., Chen, G., Povey, D., and Khudanpur, S. (2015). Librispeech: An ASR corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210, South Brisbane, Queensland, Australia. IEEE.

[54] Paulus, R., Xiong, C., and Socher, R. (2017). A Deep Reinforced Model for Abstractive Summarization.

[55] Pratap, V., Tjandra, A., Shi, B., Tomasello, P., Babu, A., Kundu, S., Elkahky, A., Ni, Z., Vyas, A., Fazel-Zarandi, M., Baevski, A., Adi, Y., Zhang, X., Hsu, W.-N., Conneau, A., and Auli, M. (2023). Scaling Speech Technology to 1,000+ Languages.

[56] Pratap, V., Xu, Q., Sriram, A., Synnaeve, G., and Collobert, R. (2020). MLS: A Large-Scale Multilingual Dataset for Speech Research. In *Interspeech 2020*, pages 2757–2761.

[57] Radford, A., Kim, J. W., Xu, T., Brockman, G., McLeavey, C., and Sutskever, I. (2022). Robust Speech Recognition via Large-Scale Weak Supervision.

[58] Ratcliff, R. (1990). Connectionist models of recognition memory: Constraints imposed by learning and forgetting functions. *Psychological Review*, 97(2):285–308.

[59] Romero, A., Ballas, N., Kahou, S. E., Chassang, A., Gatta, C., and Bengio, Y. (2014). FitNets: Hints for Thin Deep Nets.

[60] Sanh, V., Debut, L., Chaumond, J., and Wolf, T. (2020a). DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter.

[61] Sanh, V., Wolf, T., and Rush, A. M. (2020b). Movement Pruning: Adaptive Sparsity by Fine-Tuning.

[62] Schneider, S., Baevski, A., Collobert, R., and Auli, M. (2019). Wav2vec: Unsupervised Pre-training for Speech Recognition.

[63] Senin, P. (2009). Dynamic time warping algorithm review.

[64] Sennrich, R., Haddow, B., and Birch, A. (2015). Neural Machine Translation of Rare Words with Subword Units.

[65] Shao, H., Wang, W., Liu, B., Gong, X., Wang, H., and Qian, Y. (2023). Whisper-KDQ: A Lightweight Whisper via Guided Knowledge Distillation and Quantization for Efficient ASR.

[66] Shao, L., Gouws, S., Britz, D., Goldie, A., Strope, B., and Kurzweil, R. (2017). Generating High-Quality and Informative Conversation Responses with Sequence-to-Sequence Models.

[67] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention Is All You Need.

[68] Vijayakumar, A. K., Cogswell, M., Selvaraju, R. R., Sun, Q., Lee, S., Crandall, D., and Batra, D. (2016). Diverse Beam Search: Decoding Diverse Solutions from Neural Sequence Models.

[69] Voita, E., Talbot, D., Moiseev, F., Sennrich, R., and Titov, I. (2019). Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned.

[70] Wang, C., Rivière, M., Lee, A., Wu, A., Talnikar, C., Haziza, D., Williamson, M., Pino, J., and Dupoux, E. (2021). VoxPopuli: A Large-Scale Multilingual Speech Corpus for Representation Learning, Semi-Supervised Learning and Interpretation.

[71] Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T. L., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. M. (2019). HuggingFace's Transformers: State-of-the-art Natural Language Processing.

[72] Zhang, Y., Park, D. S., Han, W., Qin, J., Gulati, A., Shor, J., Jansen, A., Xu, Y., Huang, Y., Wang, S., Zhou, Z., Li, B., Ma, M., Chan, W., Yu, J., Wang, Y., Cao, L., Sim, K. C., Ramabhadran, B., Sainath, T. N., Beaufays, F., Chen, Z., Le, Q. V., Chiu, C.-C., Pang, R., and Wu, Y. (2021). BigSSL: Exploring the Frontier of Large-Scale Semi-Supervised Learning for Automatic Speech Recognition.

[73] Zhang, Y., Qin, J., Park, D. S., Han, W., Chiu, C.-C., Pang, R., Le, Q. V., and Wu, Y. (2020). Pushing the Limits of Semi-Supervised Learning for Automatic Speech Recognition.

[74] Zhu, M. and Gupta, S. (2017). To prune, or not to prune: Exploring the efficacy of pruning for model compression.

Appendix A

Evaluation datasets

This section of the appendix describes how each of the evaluation datasets for this thesis were prepared. Similarly to subsection 5.1.1, the data has been downloaded using HuggingFace's `datasets` to make the experiments easily reproducible [40].

A.1 End-to-End Speech Benchmark (ESB)

The End-to-End Speech Benchmark (ESB) evaluation dataset is a collection of 8 datasets first introduced by [20]. Details about each individual dataset are shown in Table A.1.

Dataset	Domain	Audio length (h)
AMI-IHM [9]	Meetings	9
Common Voice (English) [3]	Wikipedia	27
Earnings-22 (English) [17]	Meetings	5
GigaSpeech [11]	Audiobook, podcast, YouTube	40
LibriSpeech (clean+other) [53]	Audiobook (LibriVox)	11
SPGISpeech [52]	Meetings	100
TED-LIUM [28]	TED talks	3
VoxPopuli (English) [70]	EU Parliament	5

Table A.1 Details about the ESB dataset group.

Note that the ESB-diagnostic dataset group is an 8-hour lightweight subset of ESB that can be used for faster iterations.

A.2 ESB diagnostic custom (ESBDC)

Both ESB and ESB-diagnostic don't provide access to the test set of LibriSpeech and AMI, which are our target training sets. Therefore, we had to create a custom version of ESB-diagnostic where we properly replaced the missing test splits. For this reason, we will refer to this dataset group as ESB diagnostic custom (ESBDC). Details about each individual dataset from ESBDC are shown in Table A.2.

Split	Total duration (h)
Librispeech clean	5.4
Librispeech other	5.1
AMI	5.7
Common Voice	1
Voxpopuli	1
TED-LIUM	1
GigaSpeech	1
SPGISpeech	1
Earnings-22	1

Table A.2 Details about the ESBDC dataset group.

A.3 MultiLingual LibriSpeech (MLS)

The MultiLingual LibriSpeech (MLS) evaluation dataset used here is a collection of 8 datasets [56]. MLS consists of 8 ASR subsets: Dutch, English, French, German, Italian, Polish, Portuguese, and Spanish. Note that the original MLS dataset was slightly modified because we wanted to reuse the LibriSpeech data from Panayotov et al. [53] for consistency. Therefore, the English dataset is the result of concatenating the `librispeech_clean` and `librispeech_other` splits. Details about each individual dataset are shown in Table A.3.

Split	# examples	Total duration (h)
Dutch	3075	4.1
English	5559	10.5
French	2426	7.7
German	3394	16.1
Italian	1262	4.8
Polish	520	2.4
Portuguese	871	4.4
Spanish	2385	9.0

Table A.3 Details about the MLS dataset.

A.4 Forgetting Assessment Dataset (FAD)

The Forgetting Assessment Dataset (FAD) evaluation dataset is a custom dataset created for this project that contains a few datasets that cover different ASR tasks. Although it is redundant with the previous dataset groups, we decided to create it for easier evaluation to easily evaluate forgetting relative to fine-tuning with respect to the number of training steps. Details about each individual dataset are shown in Table A.4.

Split	Total duration (h)	Description
AMI	5.7	The AMI test set (see section 34)
TED-LIUM	1.0	The TED-LIUM test set from ESBDC (see section A.2)
MLS French	7.7	The French test set from MLS (see section A.3)

Table A.4 Details about the FAD dataset.