ALMA MATER STUDIORUM · UNIVERSITÀ DI BOLOGNA

DEPARTMENT OF COMPUTER SCIENCE

AND ENGINEERING

ARTIFICIAL INTELLIGENCE

MASTER THESIS

in

Natural Language Processing

ON THE USE OF PROMPTING FOR FINE-TUNING NEURAL MODELS FOR SPEECH PROCESSING

CANDIDATE

Stefano Ciapponi

SUPERVISOR

Prof. Paolo Torroni

CO-SUPERVISOR

Alessio Brutti, PhD.

Academic year 2022-2023

Session 2nd

Dedicata ad amici e famiglia, persone e animali,
per essermi stati vicino durante questi (due) anni a Bologna.
Non sto a fare tutti i nomi, se no la lista non finisce più.

Abstract

Recent advances in the development of extremely large, multi-purpose models have motivated computer scientists to explore methods for adapting them to more specific tasks. Fine-tuning is the most widely used approach to this problem, in which a more general model is trained on a new dataset of labeled data for the new task. While fine-tuning mitigates the data availability problem and enables models trained on small labeled datasets to achieve state-of-the-art performance, it also exhibits some key disadvantages: inefficiency, resource-intensive computation and making the models less general. This study investigates the use of *learnable prompts*, a parameter-efficient fine-tuning alternative, in spoken language understanding (SLU) tasks. To our knowledge, learnable prompts have not been previously applied to SLU, but have been tested on text-based natural language processing (NLP) tasks and computer vision tasks, achieving promising results. Therefore, we'll be introducing our proposed approach, using learnable prompts in a SLU context, and analyse some experimental results on two different deep learning-based end-to-end SLU models.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Recent advances in the development of extremely large, multi-purpose models have motivated computer scientists to explore methods for adapting them to more specific tasks, which often lack adequate labeled datasets. Standard Fine-tuning is the most widely used approach to this problem, in which a more general model, often pre-trained on semi-supervised tasks, is trained on a new dataset of labeled data for the new task, modifying all the model weights with backpropagation. While fine-tuning mitigates the data availability problem and enables models trained on small labeled datasets to achieve state-of-the-art performance, it also exhibits some key disadvantages. Modifying all of the model weights is typically inefficient, compared to modifying only a subset of them; to some extent resource intensive, since it multiple versions of a model to respond to different tasks; and reduces the underlying model's generalization capabilities, since all the model weights are changed at training time, making the model specialized in a more restricted number of tasks.

This study investigates the use of *learnable prompts*, a parameter-efficient fine-tuning alternative, in spoken language understanding (SLU) tasks. To our knowledge, learnable prompts have not been previously applied to SLU, but have been tested on text-based natural language processing (NLP) tasks and computer vision tasks, achieving promising results. Therefore, we'll be introducing our proposed approach, using *learnable prompts* in an SLU context, and analyse some experimental results on two different deep Learning based End to End SLU models.

SLU is the field of computer science that deals with the ability of machines to understand the meaning of spoken language, It is a broad area which happened to see some paradigm changes during time. For this reason, in chapter 2, we'll start by introducing some concepts and historical overview about fine-tuning alternatives and SLU models. This will help us contextualise our proposed approach in chapter 3, based on end-to-end SLU foundation models, namely *Audio Spectrogram Transformers* and *Wav2Vec 2.0* and two different *learnable prompt* techniques as a fine-tuning alternative. In chapter 4 we'll describe our experimental setup, the task we're dealing with and report the experimental results on both models, then, in chapter 5 we'll confront the results from both models and reason around the way they learn new tasks using learnable prompts. Finally, in chapter 6 we'll draw conclusions and think about some future improvements and SLU application domains in which *learnable prompts* and parameter-efficient fine-tuning could be applied.

Chapter 2

State of the art

This chapter will review the evolution of automatic speech recognition (ASR) from statistical based to end-to-end deep learning approaches, its application in SLU systems, and discuss alternatives to fine-tuning neural models. We will begin by providing a brief overview of the traditional acoustic-language model paradigm. Next, we will discuss the recent shift towards end-to-end SLU systems, and highlight the key advances that have been made in this area. Finally, we will discuss alternatives to fine-tuning neural models, and consider their potential impact on the future of ASR and SLU.

2.1 Historical Overview, from ASR to SLU

Automatic Speech Recognition (ASR) is the process of converting spoken language into written text. It is a field of computer science that has been developing for many years, and has made significant progress in recent years. Some examples of areas which employ ASR in use today are voice assistants, smart speakers, transcription services, video captioning services and voice control systems. Generally, ASR systems work in three steps:

- First, they extract acoustic features from an audio signal.
- These features are then used to train machine learning models to recognize

different words and phrases.

- Once the model is trained it can be used to decode new audio signals and, typically, generate text transcripts.

One step further into the application of ASR is Spoken Language Understanding (SLU). Academically speaking, SLU can be defined as the task of automatically extracting meaning from spoken language. Once the speech has been transcribed or processed by an ASR system, a SLU system uses a variety of NLP techniques to understand the meaning of the spoken utterances. These techniques may include:

- Intent detection: Identifying the user's goal or intention in speaking.
- Slot filling: Extracting specific information from the user's speech.
- Entity recognition: Identifying named entities in the user's speech, such as people, places, and organizations.
- Dialogue state tracking: Keeping track of the state of a conversation in order to provide more relevant and informative responses.

Once the SLU system has understood the meaning of the user's speech, it can then generate an appropriate response. To sum everything up: ASR is the process of converting spoken language into text, while SLU is the process of understanding the meaning of that text. ASR is a prerequisite for SLU, but SLU is more complex than ASR because it requires the system to understand the context of the speech and the speaker's intentions.

The evolution of ASR systems, often employed for SLU tasks, was subject to a paradigm shift, from a statistical based approach to a more data-driven approach, which lead to the implementation of end-to-end SLU systems, in which the acoustic and language models, part of the SLU system, are not modeled distinctively. We'll showcase this paradigm shift more in detail in the next paragraphs.

Hidden Markov Models Hidden Markov models (HMMs) [1] are a powerful tool for modeling sequential data, such as speech signals. They can capture the probabilistic dependencies between the observed features and the underlying states of a system, and allow for efficient inference and learning algorithms. In the context of speech recognition, an HMM can be used to model the sequence of acoustic features that are extracted from a speech signal. The hidden states of the HMM represent the different phonemes or words in the speech signal, and the observations represent the acoustic features.

The HMM assumes that the speech signal evolves over time according to a Markov process, meaning that the current state depends only on the previous state. This is a reasonable assumption for speech signals, as the acoustic features of a phoneme or word are typically influenced by the preceding phoneme or word. HMMs are trained on a corpus of speech data, which is used to estimate the probabilities of the hidden states and the observations. Once the HMM is trained, it can be used to recognize speech by finding the most likely sequence of hidden states that generated the observed acoustic features.

Studies on the applications of HMMs to speech recognition have been documented as early as 1986 in Bahl et Al. paper *Maximum mutual information estimation of hidden Markov model parameters for speech recognition* [2]. In this article IBM researchers propose a method for estimating the parameters of hidden Markov models of speech, in which parameter values are chosen to maximize the mutual information between an acoustic observation sequence and the corresponding word sequence.

In 1991 Bell Lab researchers, Juang et Al. wrote an expository article, *Hidden Markov Models for Speech Recognition* [3], a comprehensive overview of HMMs applied in a Speech Recognition domain in which they show how the strengths of the method lie in the consistent statistical framework that is flexible and versatile and that and the ease of implementation makes the method practically attractive, showing that HMM systems were capable of achieving recogntiion rates of more than 95% word accuracy in certain speaker-independent tasks, with vocabularies

on the order of 1000 words.

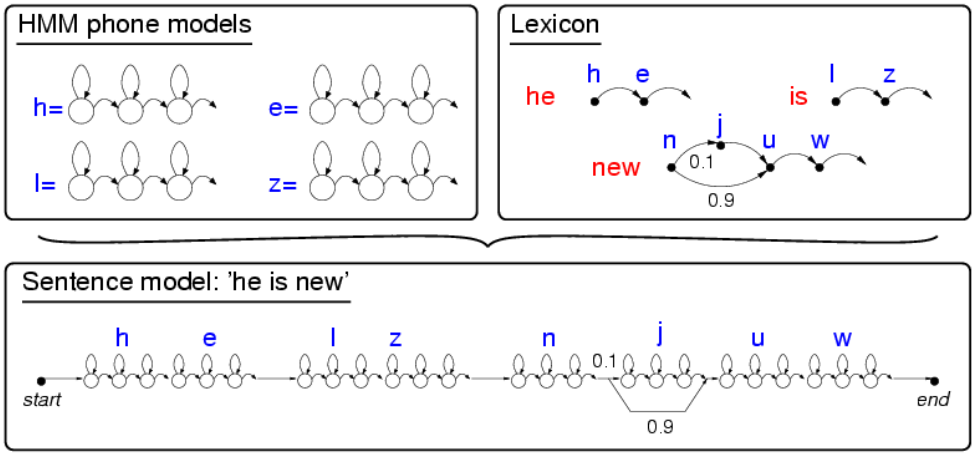

Figure 2.1: HMM for sentence modeling diagram (esat.kuleuven.be)

In 2007 Gales et Al. published *The Application of Hidden Markov Models in Speech Recognition* [4] a review of the core architecture of an HMM-based system and a case study on Large Vocabulary Continuous Speech Recognition (LVCSR) for broadcast News and Conversation Transcription. This is done by using an acoustic model to compute the *Likelihood* of a word alongside a language model to compute the *prior* of the given word in a sentence structure. The researchers conclude the review pointing out that the HMM architecture results in transcription tasks fall short of human capabilities and that it is arguable that the HMM architecture is fundamentally flawed and that performance must asymptote at some point, but no good alternatives to the HMM were found yet. The limits researchers wrote about consisted in three key points: HMMs are not able to learn long-range dependencies in the data, they are sensitive to the choice of features and are computationally more expensive than other models, which can make them impractical for some real-time applications. The shortcomings of the hidden Markov model approach to automatic speech recognition were overcome by the development of deep learning ASR models.

Deep Learning-based ASR and End to End SLU Deep learning [5] has revolutionized the field of SLU in recent years. Deep learning models are able to learn

complex relationships between speech features and text transcripts, which leads to more accurate and robust SLU systems.

One of the first articles about the use of Deep Learning in an ASR context is *Acoustic Modeling Using Deep Belief Networks* by Mohamed et Al. [6] (2011), in which researchers train a Deep Belief Network as a replacement to Gaussian-mixture model to estimate the emission distribution model for an HMM. This is done through a multi-layer generative model of a window of spectral feature vectors without making use of any discriminative information.

A step forward was made in 2012 after the publishing of Abdel-Hamid et Al. article *Applying Convolutional Neural Networks concepts to hybrid NN-HMM model for speech recognition* [7]. In this paper researchers applying CNN principles along the frequency axis of speech signals because the variability along the time axis is handled by the HMM model and the dependency between adjacent speech frames is dealt with a long time context window feeding as an input to the NN as in standard hybrid NN-HMM models [6].

Another important progress in the field, which happened during the last decade, was the introduction of *End to End (E2E)* SLU models. What differentiates them from the typical HMM-based model is the ability of learning the mapping from speech signals to text transcripts directly, without the need for any intermediate steps, such as acoustic modeling or language modeling. Most end-to-end speech recognition models include the following parts: encoder, which maps speech input sequence to feature sequence; aligner, which realizes the alignment between feature sequence and language; decoder, which decodes the final identification result. One thing to note is that this division does not always exist, because end-to-end itself is a complete structure, and it is usually very difficult to tell which part does which sub-task [8].

Table 2.1 displays a comparison between end-to-end architectures and previous approaches. Some of the reasons which E2E architectures to be conceived are:

- *Advances in Deep Learning*, which revolutionised ASR alongside many other

Criterion	E2E SLU	Previous approaches
Training:	Trains a single model to directly map speech signals to text transcripts.	Trains two separate models: an acoustic model and a language model.
Deployment:	Deploy a single model.	Deploy two separate models
Accuracy:	State-of-the-art accuracy on a variety of benchmarks.	Good accuracy on a variety of benchmarks, but not as good as E2E ASR.
Robustness:	More robust to noise and other distortions.	Less robust to noise and other distortions.
Data requirements:	Requires large amounts of labeled speech data to train.	Requires less labeled speech data to train than E2E ASR, but still requires a significant amount.
Computation cost:	Can be computationally expensive to train and deploy.	Can be less computationally expensive to train and deploy than E2E ASR, but still requires a significant amount of computation.

Table 2.1: Comparison between E2E ASR and HMM

fields in recent years;

- *Availability of large datasets*, since E2E SLU systems require large amounts of labeled speech data to train;
- *Increased Computational Resources*, because E2E models are usually extremely expensive computationally

One of the main Advances in Deep Learning which lead to the development of E2E models is the invention of the *Connectionist Temporal Classification (CTC) loss function*, proposed by *Graves et Al.* [9] in 2006. CTC loss works by considering all possible alignments between the input audio sequence and the output text sequence. To do this, it allows the output sequence to be repeated or blank symbols to be inserted. The CTC loss is then calculated as the negative log-likelihood of the correct alignment, given the input sequence. To understand how CTC loss works, it is helpful to first understand the concept of a *lattice*. A lattice is a *directed acyclic graph* (DAG) that represents all possible alignments between two sequences. Each node in the lattice represents a possible state of the alignment, and each edge represents a transition from one state to another.

The CTC loss is calculated using the *forward-backward algorithm* [10], an inference algorithm for hidden markov models (HMM). This algorithm computes the probability of each state in the lattice, given the input sequence. The CTC loss is then calculated as the negative log-likelihood [11] of the state that corresponds to the correct alignment.

One of the first applications of CTC to large vocabulary speech recognition was documented in *Towards End-to-End Speech Recognition with Recurrent Neural Networks* by Graves et al. in 2014 [12]. They combined a hybrid DNN-HMM and a CTC trained model to achieve state-of-the-art results [13]. In this paper *Graves et Al.* propose a system is based on a combination of the deep bidirectional LSTM [14], [15] recurrent neural network architecture and the connectionist Temporal Classification objective function. Through this paper they managed to demonstrate that character-level speech transcription can be performed by a recurrent neural network with minimal preprocessing and no explicit phonetic representation.

The most recent breakthrough in the SLU field is the adoption of Transformer Models [16], a type of neural network architecture that have become the state-of-the-art for many natural language processing (NLP) (such as machine translation, text summarization, and question answering) and Computer Vision tasks. These models are based on the self-attention [17] mechanism, which allows them to learn long-range dependencies in sequential data without the need for recurrent connections. Self-attention allows a transformer to learn how each element in a sequence relates to all other elements in the sequence, regardless of their distance apart. This is in contrast to recurrent neural networks (RNNs), which can have difficulty learning long-range dependencies because of the vanishing gradient problem [18].

The first documented application of Transformer neural networks to SLU tasks was *Speech-Transformer: A No-Recurrence Sequence-to-Sequence Model for Speech Recognition*, Dong et Al. 2018. In this paper researchers present a no-recurrence sequence-to-sequence model entirely relies on attention mechanisms to learn the positional dependencies, which can be trained faster with more efficiency than recurrent sequence-to-sequence models. Their best model achieves competitive word error rate (WER) of 10.9%, while the whole training process only takes 1.2 days on 1 GPU, significantly faster than the published results of recurrent sequence-to-sequence models.

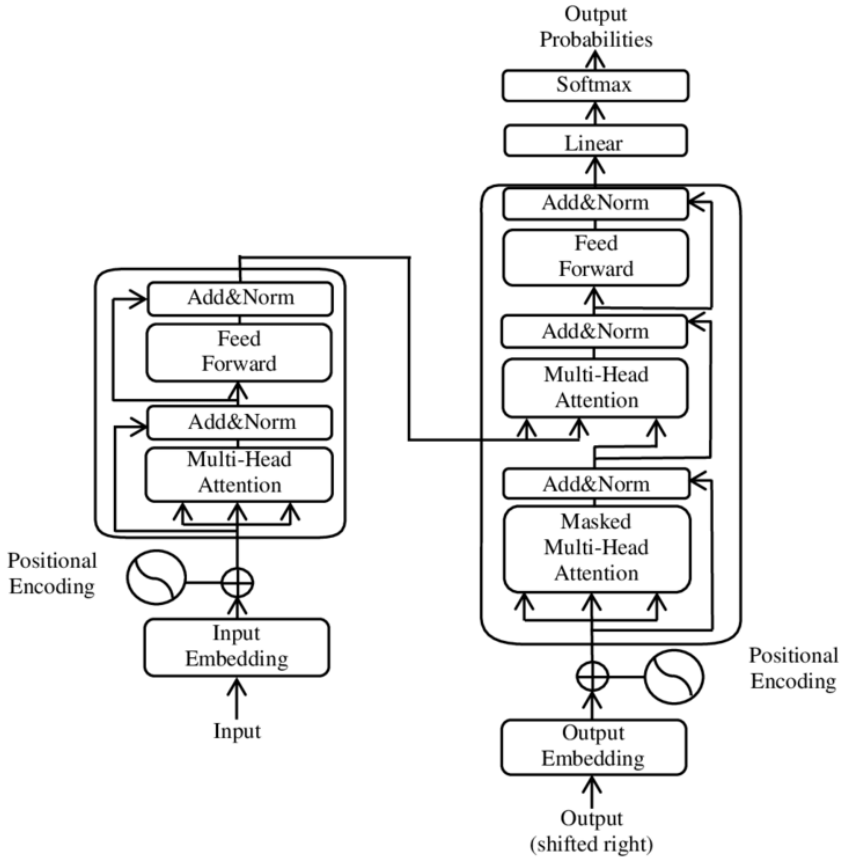

Figure 2.2: Transformer Model Architecture (Encoder-Decoder)

The advantage of the adoption of Transformer models, compared to Recurrent Neural networks has been confirmed by Karita et Al. paper, *A Comparative Study on Transformer vs RNN in Speech Applications* [19] in 2019. The study results show that Transformer outperforms RNN on 13/15 corpora in their experiment. Although their system has no pronunciation dictionary, part-of-speech tag nor alignment-based data cleaning unlike Kaldi [20], an HMM ASR tolkit, their Transformer provides comparable CER/WERs to the HMM-based system, Kaldi on 7/12 corpora.

In 2019 studies about *A new joint CTC-attention-based speech recognition model with multi-level multi-head attention* [21] were conducted by Qin et. Al. This lead to a series of articles improving ASR and SLU models, utilizing transformer-based

and transformer-hybrid technologies. In the future Automated Speech Recognition technologies are expected to become more and more pervasive, accurate and available to everyone. To do that the scientific community is both researching ways to improve the models accuracy, but also ways to ease model deployment, retraining and make models run on heterogeneous hardware, overcoming the limits of Transformer-based models [22]. For this reason we decided to run some tests using a parameter-efficient fine-tuning alternative on AST models, which could lead to easier deployement and ease model retraining.

2.2 Fine-tuning Alternatives

Fine-tuning is a common technique for adapting pre-trained models to new tasks. It involves updating *all of the model parameters* to minimize the loss on a labeled dataset for the target task. However, fine-tuning can be computationally expensive and can lead to overfitting, especially for large pre-trained models. *Adapter-tuning* and *Prompt-Tuning* are two emerging parameter-efficient techniques for adapting pre-trained models (PMs) to new tasks with fewer parameters and less training time. Some examples of fine-tuning applied to ASR and SLU models are [23]–[26].

Adapters Adapter tuning is a fine-tuning technique for large models that involves adding small, task-specific "adapter" modules to the pre-trained model. These adapters are trained on a few examples of the specific task, and can significantly improve the model's performance on that task without requiring the entire model to be retrained. Adapter tuning is based on the idea that different tasks require different types of knowledge. For example, a task like machine translation may require knowledge of different languages and cultures, while a task like question answering may require knowledge of different types of facts and relationships. By adding adapters that are specifically tailored to each task, we can help the LLM to learn the knowledge that it needs to perform that task well. Given a Transformer model with

parameters θ, Adapter-tuning inserts some task-specific modules with parameters ϕ between its layers; these modules are called *adapters*. Typically the new parameters ϕ are trained on the target task while keeping the Pre-trained model frozen, which implies that the new weights can learn to encode task-specific features [27]. It has been experimentally proven that adding a two-layer bottleneck feed-forward neural network at every layer of the pre-trained Transformer works well [28]; however, choosing the exact placement of these extra parameters is a difficult task, as it can have a significant impact on performance.

Adapter tuning has been shown to be effective for a wide range of tasks in the application domains of Natural language Processing [29], Computer Vision [30]–[32], Automatic Speech Recognition [33], [34] and Multimodal Information Retrieval [35]. It is a particularly promising technique for tasks where there is limited labeled data available, as it allows us to leverage the knowledge that the model has already learned from its pre-training.

Prompt Tuning Prompting is a newer technique that can be used as an alternative to fine-tuning. It involves providing the pre-trained network with a prompt, which is a piece of text or code that is added, appended or prepended to the network layers input and helps it understand the task and generate the desired output. One advantage of prompting is that it does not require any fine-tuning of the network's parameters. This means that it can be much faster and more efficient than fine-tuning, especially for tasks where there is only a small amount of labeled data available. Another advantage of prompting is that it can be used to adapt the network to a wide variety of tasks, even tasks that are very different from the task that the network was originally pre-trained on. There are two main types of prompts: hard prompts and soft prompts.

Hard prompts [36] are fixed pieces of text or code that are provided to the network as input. They are typically used to specify the task that the network should perform and to provide the network with some context about the input data.

Soft prompts are *learnable parameters* that are optimized during the training

process and can be seen as *virtual tokens* [37]. They are typically used to fine-tune the network's response to the prompt. *Learnable* prompts work by allowing the network to learn how to best use the prompt to generate the desired output. Learnable prompts can be used to improve the performance of prompting on a variety of tasks and they have been successfully used in different application domains, such as NLP [38][39], computer vision [40] and continual learning [41]. *Leister et Al.* [38] have also empirically shown that soft-prompting is comparable to fine-tuning in models made out of Billions of parameters.

Feature	**Soft prompt**	**Hard prompt**
Definition	A learnable parameter that is optimized during the training process.	A fixed piece of text or code that is provided to the network as input.
Interpretability	Not interpretable.	Interpretable.
Transferability	Poor transferability.	Good transferability.
Data requirements	Requires more data to train.	Requires less data to train.
Example	*[PROMPT_EMBEDDING]*	*"Translate the following sentence into Spanish: I love cats."*

Table 2.2: Comparison between Hard and Soft prompts

Table 2.2 shows the main feature differences between hard and soft prompts. Overall we could say that *learnable* soft prompts appear to be worse than *handcrafted* hard prompts under every aspect, but we should keep in mind that it's really hard to think of a way to handcraft prompts outside of a typical Text-based NLP domain. This is especially true in the context of SLU models: because of the probabilistic nature of speech, defining some fixed tokens which can be inputed by the user while interacting with the model, could be seen as a task itself, which would make hard prompting extremely complicated in this application domain.

We can see that Hard prompts are *Interpretable* since they are handcrafted and can be used to infer an expected behaviour from a neural model, while Soft learnable prompts aren't, since they are vectors of learnt parameters which have no direct meaning embodied in them. In the Table context *Transferability* [42] refers to the ability of a prompt to be used for a task that is different from the task that the prompt was trained on. For example, in a Natural Language Processing Context, a prompt that was trained for a text classification task may be transferable to

a text generation task. Although soft prompts are generally less transferable than hard prompts, since they are optimized to perform a specific task with a specific language model, they have been proven to be transferable both cross-model and cross-task [37]. This could have important applications in model deployment and in multi-model federated learning [43] settings. This could be extremely useful since, a typical use case for SLU model consists in them to be deployed directly on devices which, sometimes, collaborate together for acoustic scene understanding tasks. Soft prompting could also allow foundation models to be used on different tasks, simply changing prompt weights, without putting pressure on the server-side storage capabilities. This approach can be extremely useful in a more back-end oriented SLU use case, such as the current voice assistant products (e.g. Amazon Alexa).

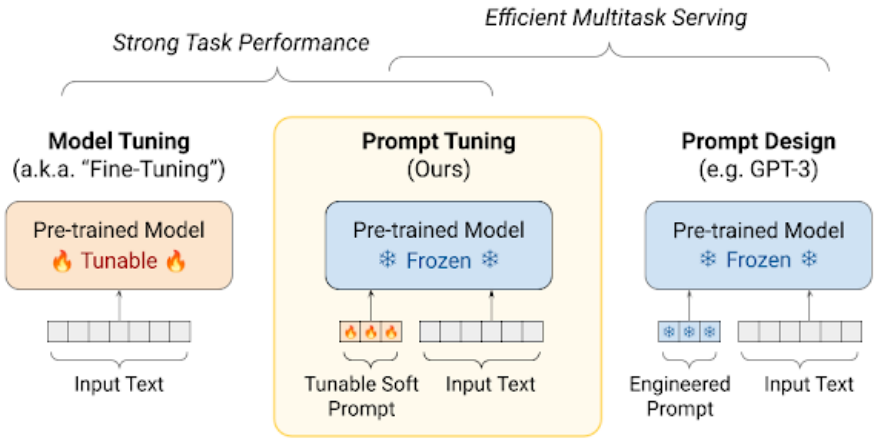

Figure 2.3: Comparison between Model Tuning, Prompt Tuning (soft prompts) and Prompt Design (hard prompts), Google Research [38]

Chapter 3

Proposed Approach

In this chapter we'll introduce the main neural architecture components which have been used and combined to test *learnable prompts* on speech. To do that we decided to employ a two transformer models: Audio Spectrogram transformers and Wav2Vec2. The former is pre-trained on Sound classification, while the latter is pre-trained identifying the true quantized latent speech representation for a masked time step within a set of distractors. We decided to test prompting techniques to fine-tune them on Automatic Speech Recognition Tasks (Speech Command Intent Classification). The chapter is divided in three main sections, namely:

- First, we'll introduce Shallow and Deep Prompting techniques originally applied both in Computer Vision and Text-based Natural Language Processing Tasks;
- Next, we'll investigate and justify the Transformer Architecture we chose to deal with Automatic Speech Recognition Tasks;
- Finally, we'll show how those two have been used together to tackle the task at hand.

This chapter will serve as an introduction to the actual experiments which will be further explored in the next chapter.

3.1 Shallow and Deep Prompting Techniques

Jia Et Al. [40] paper, *Visual Prompt Tuning*, has been our main source of inspiration for our experiments, since it revolves around an in-depth analysis of prompting techniques tested on Vision Transformers (ViT, [44]) and it caught our attention since, as of 2023, it has been one of the few examples of prompting techniques applied outside the typical Text-Based Large Language Model domain.

The authors' main focus of research is finding ways to reduce the memory impact and, consequentially, ease deployment of Fine Tuned Transformer-based Vision models. To do that they employ two Prompting Techniques, namely: *VPT-Shallow* and *VPT-Deep*, whose main difference is the number of Transformer layers involved in the task.

General Overview Given a Transformer Model (ViT in [40] case) with N layers, an input vector of m input features $I_j, 1 \leq j \leq m$ is mapped in a collection of input embeddings $e_0^j, j = 1, 2, ..., m$, each computed by en embedding function $Embed(I_j)$, into in a d-dimensional latent space with positional encoding.

This collection is denoted as $\mathbf{E}_i = \{e_i^j \in \mathbb{R}^d | j \in \mathbb{N}, 1 \leq j \leq m\}$ and represents the inputs of the $(i+1)$-th Transformer layer (L_{i+1}). Together with $x_i \in \mathbb{R}^d$, an extra learnable classification token ([CLS]), the whole transformer model is formulated as:

$$[x_i, E_i] = L_i([x_{i-1}, E_{i-1}])i = 2, 3, ..., N \tag{3.1}$$

$$y = \textsc{Head}(x_N) \tag{3.2}$$

Shallow Prompt Tuning Prompts are inserted into the first Transformer layer L_1 *only*.

A collection of p prompts is denoted as $P = \{p^k \in \mathbb{R}^d | k \in \mathbb{N}, 1 \leq k \leq p\}$.

Given that, a shallow-prompted Transformer can be defined as:

$$[x_1, Z_1, E_1] = L_1([x_0, \mathbf{P}, E_0]) \tag{3.3}$$

$$[x_i, Z_i, E_i] = L_i([x_{i-1}, Z_{i-1}, E_{i-1}]), \quad i = 2, 3, ..., N \tag{3.4}$$

$$y = \text{Head}(x_N) \tag{3.5}$$

where $\mathbf{Z}_i \in \mathbb{R}^{p \times d}$ represents a set of features computed by the i-th Transformer layer , and $[x_i, Z_i, E_i] \in \mathbb{R}^{(1+p+m) \times d}$. In this approach the only learnable parameters of the model are P, the set of prompts, and $Head$, the Multi-Layer-Perceptron Classification Head. This results in an extremely small amount of weight changes compared to the original model (around 1-2% of the total model weights), whose weights remain frozen at training time.

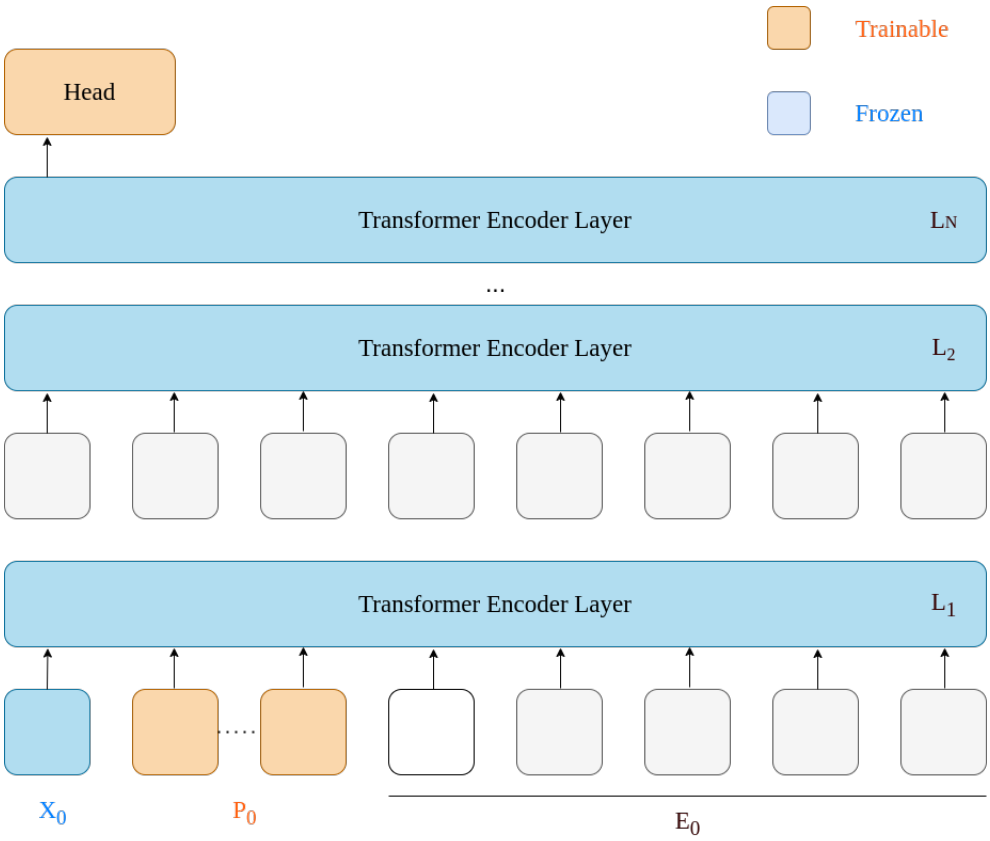

Figure 3.1: SPT, Architecture Scheme

In figure 3.1 we can see a brief architecture scheme for the SPT approach, highlighting which components of the network architecture are trainable and which are not.

Deep Prompt Tuning Prompts are injected in *every* Transformer layer's input space. In the context of a Transformer Model with N layers, we define N collections of input learnable prompts: $\mathbf{P}_i = \{p_i^k \in \mathbb{R}^d | k \in \mathbb{N}, 1 \leq k \leq m\}, i = 1, 2, ..., N$.

The deep-prompted transformer is the formulated as:

$$[x_i, \_, E_i] = L_i([x_{i-1}, P_{i-1}, E_{i-1}]), \quad i = 2, 3, ..., N \tag{3.6}$$

$$y = \textsc{Head}(x_N) \tag{3.7}$$

As we can derive from the previous equations, prompts collections are learnt and then, at each forward propagation step, a prompt-set P_i is replaced by a prompt set P_{i+1}, this results in each layer L_i output having the same dimension: $[x_i, \_, E_i] \in \mathbb{R}^{(1+p+m)\times d}$.

In figure 3.2 we can see an architecture scheme similar to figure 3.1. The main difference between the two approaches is highlited by the *trainable parameters*, which, in the former approach are appended to each encoder input. This leads to two different ways of thinking about the prompt size p.

Either:

- p is the total number of prompts, so each layer is assigned $p/12$ prompts.
- or, p is the prompt number for *each layer*, so the model is learning $12 \times p$ prompts.

During the implementation we decided to adopt the latter method, therefore if we are testing the *DPT* approach with 50 tokens, we are actually learning a prompt pool of $12 \times 50 = 600$ tokens, throughout all the Transformer Layers.

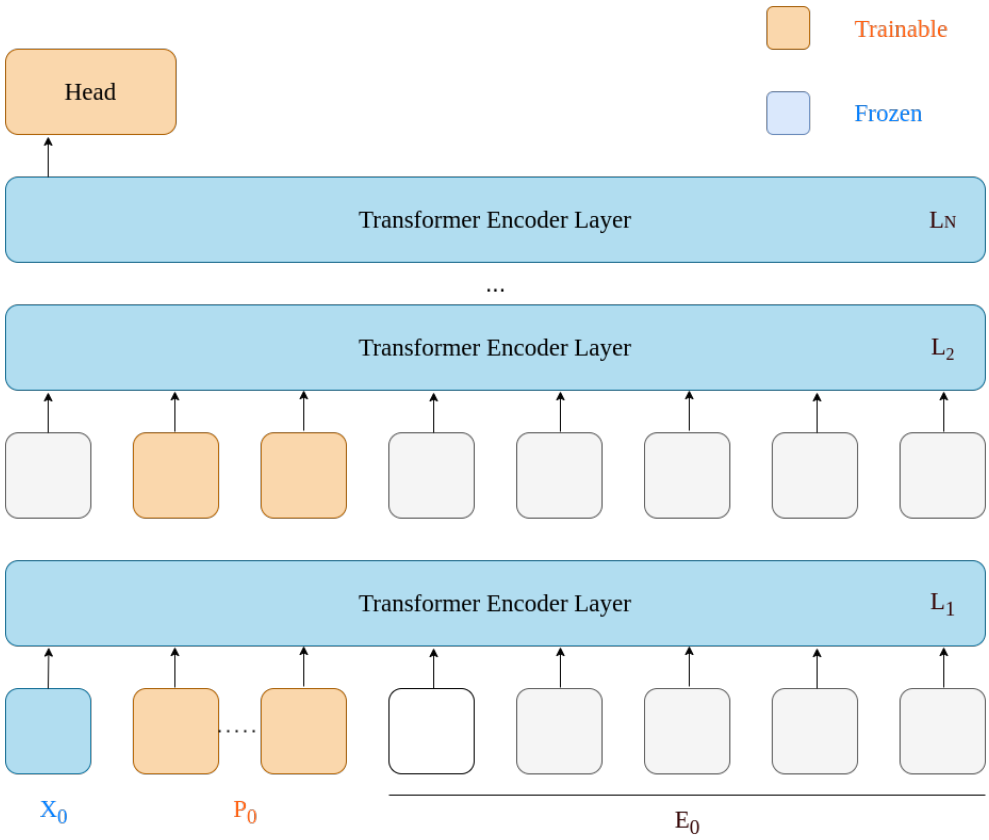

Figure 3.2: DPT Architecture Scheme

3.2 Audio Spectrogram Transformer

Since Jia Et Al. paper is based on Vision Transformers [44] we decided to work with a similar architecture trained for *Sound Classification* tasks. *Audio Spectrogram Transfomers (AST)*, introduced by Gong et Al. [45] in 2022, tackle Audio Classification Problems using a convolution-free, purely attention-based model. This architecture was developed to understand whether is it possible get rid of convolutions in a CNN-Attention hybrid Audio Classification Neural Architecture achieving comparable performance. To deal with this challenge they decided to modify a standard Vision Transformer Architecture so that it accepts *Spectrograms* as input and deals with Sound Classification instead of the standard Image Classification task.

Audio Pre-processing As the name suggests an *Audio Spectrogram Transformer* uses Spectrograms as an input to the Transformer model. Spectrograms can be defined as the visual representation of frequencies in a signal, as it varies with time. They can be seen as a bi-dimensional feature map in which one axis represents time while the other axis represents frequency; each pixel intensity represents the amplitude of a particular frequency at a particular time (usually discretised through frequency binning and time windowing). Spectrograms may be created from a time-domain signal in one of two ways: either approximated as a filterbank that results from a series of band-pass filters or calculated from the time signal using a Fourier Transform (typically Short-Time Fourier Transform). Architectures based on Spectrograms have been employed since the early days of experiments in the Neural Network Domain, an example of such is *Yeshwant et Al.* [46] in which spectrograms are compared to cochleagrams (similar feature map which mimics the outer- and middle-ear response to frequencies) and appear to share a similar performance when used as an input for a vowel classification problem.

In the AST context, inputs are pre-processed the following way:

1. First, the input audio waveform of t seconds is converted into a sequence of *128*-dimensional log Mel filterbank (fbank) features, computed with a 25ms Hamming window every 10ms.

   This results in a $128 \times 100t$ spectrogram as input to the Audio Spectrogram Transformer.

2. The Spectrogram is divided into a sequence of N 16×16 patches with an overlap of 6 both in time and frequency dimension.

   $N = 12[(100t - 16)/10]$ is the number of patches and the effective input sequence length for the Transformer.

3. Each 16×16 patch is flattened into a 1D patch of size 768 using a *linear projection layer* (patch embedding).

4. A trainable positional embedding (size 768) is added to capture the structure

of the 2D audio spectrogram.

The *patch embedding layer* can be viewed as a single convolution layer with large kernel and stride size and the projection layer in each Transformer block is equivalent to 1×1 convolution. Figure 3.3 shows diagram displaying the pre-processing steps.

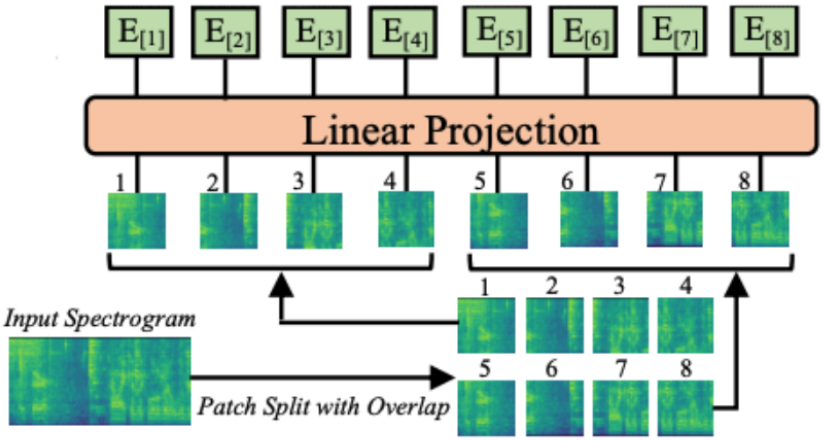

Figure 3.3: AST Preprocessing & Patch Embedding, diagram from the original paper

Cross-Modality Transfer Learning The paper's authors decided to employ *cross-modality transfer learning* from a Vision Transformer model trained on Imagenet. This is motivated by the lack of large audio datasets, which are typically of smaller size compared to their Image counterpart. To do that they employed a few modifications to the standard ViT architecture, namely:

- Since AST uses 1-channel input Spectrograms, compared to ViT, which uses 3-channel input images, the weights corresponding to each of the three input channels are averaged, which is equivalent to expanding a 1-channel input to 3-channels, but more computationally efficient.

- The authors propose a cut and bi-linear interpolation method for positional embedding adaptation. This is due to Spectrograms having variable length compared to the fixed Image size of Vision Transformers.

Label	Quality estimate?	Number of videos
Music	100%	1,011,305
Speech	100%	1,010,480
Vehicle	100%	128,051
Musical instrument	100%	117,343
Plucked string instrument	100%	44,565
Singing	100%	42,493
Car	100%	41,554
Animal	100%	40,758

Figure 3.4: Some of the audioset labels

These modifications allowed them to transfer 2D spatial Knowledge from a pre-trained ViT to AST taking into account differences in the input shape.

The model has then been Trained on AudioSet, a collection of over 2 million 10 second audio clips excised from Youtube videos and labeled with the sounds that the clip contains from a set of 527 labels (figure 3.4). Overall the model is able to achieve around 98% accuracy when fine-tuned on the Google Speech Command V2 dataset [47]. This is promising because the dataset we are interested in testing the model with has a similar structure.

3.3 Wav2Vec 2.0

We decided to apply some further investigations on a different foundation model for Automatic Speech Recognition to see whether our proposed approach works on a wider range of models. To do that we decided to employ *Wav2Vec 2.0* [48] (Baevski ed Al, 2020), another model based on a Transformer architecture, whose input features differ from AST.

Input Features Instead of receiving spectrograms as an input for the AST model, Wav2Vec takes *Raw Waveforms* as an input. These waveforms are processed the following way:

1. The signal (speech) passes through 7 Convolution layers, namely:

   - A $1-d$ convolution layer with 1 input channel, 512 output channels, kernel size 10, stride 5 a GELU activation function and a normalization layer.
   - $4 \times 1 - d$ convolution layers with 512 input channel, 512 output channels, kernel size 3, stride 2 and a GELU activation function.
   - $2 \times 1 - d$ convolution layers with 512 input channel, 512 output channels, kernel size 2, stride 2 and a gelu activation function.

   The effective window size and stride of the encoder are 400 and 320, respectively. For example, for 1-second input, the encoder outputs 49 (temporal) × 512 (feature) vector. [49].

2. The $t \times 512$ feature vectors are projected into a $t \times 768$ vector through a feature projection layer (MLP).

3. Each $t \times 768$ vector is used as an input to a 12-layer transformer encoder with convolutional positional embedding and pre-trained to distinguish true future audio samples from fake distractor samples by using the context vector (transformer output). This is done by solving a contrastive task, identifying a true quantized latent speech representation for a masked time step within a set of distractor, augmented by a diversity loss, encouraging the model to use the negative and positive examples equally often.

Pretraining Dataset Differently to Audio Spectrogram Transformer, Wav2Vec is pretrained on 960 hours of Raw Speech. Wav2vec is trained using a self-supervised approach called contrastive learning. In contrastive learning, the model is presented with pairs of audio clips, one original and one modified, and is tasked with distinguishing between the two. The model learns to do this by extracting features from the audio clips that are robust to the modifications, such as background noise and speaker variations. This leads to better performances on Speech Classification,

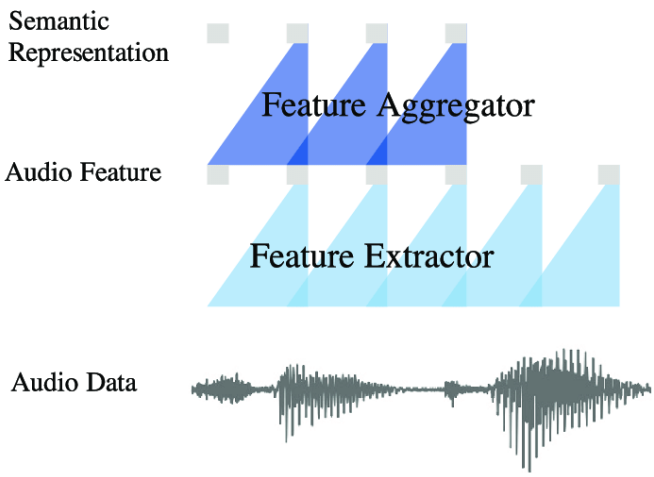

Figure 3.5: Simplified view of the Wav2Vec Backbone

such as 99.7% on Fluent Speech Command End-to-End intent Classification [50]. Because of this reason we expect the model to achieve better results when tested on learning the same Speech Recognition Task.

3.4 Prompting Implementation

Combining Audio Spectrogram Transformers and Wav2Vec with Shallow and Deep Prompt-Tuning Techniques is quite straightforward, given that these techniques are generalizable to any Transformer-Based Model. Prompts are simply declared as Learnable Parameters, which are prepended after the *Linear Projection Layer* (SPT case) or at *each* transformer layer ([DTP] case) over the AST model.

Figure 3.6 illustrates prompting added to an Audio Spectrogram Transformer Architecture, for SPT. The DPT diagram is analogous, except the backbone part on top of the linear projection layer should be replaced by figure 3.2. We decided to divide the spectrogram into 6 patches to make the diagam more understandable, in actuality all spectrograms are divided into 16×16 patches, similarly to the original Vision Transformer paper [44]. Each prompt is defined as a learnable parameter, initialized with a Xavier Uniform Initializer.

Similarily, figure 3.7 shows SPT prompting used on the Wav2Vec Architecture.

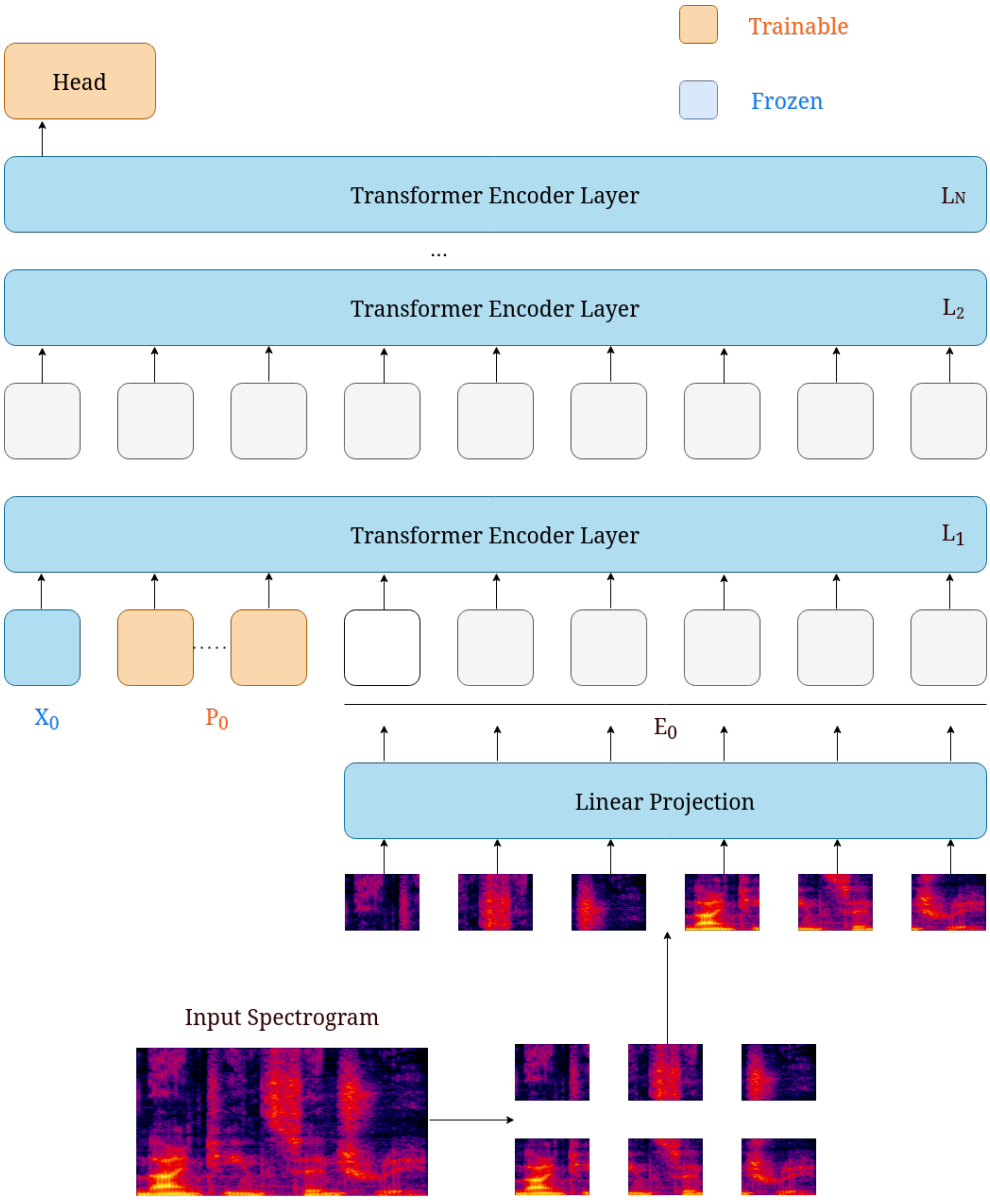

Figure 3.6: SPT, Prompted Architecture Diagram, AST (3×2 patches example)

The audio embedding part is also simplified in this diagram, since the input audio usually produces a larger amount of temporal phoneme embeddings. One key difference between this prompting scheme and the one on AST is that we prepend prompts *after* the convolutional front-end, this means that our virtual tokens represent higher level features of raw speech compared to the ones we are adding to the AST input, which is only a linear projection of the spectrogram patches.

To conclude, table 3.4 contains a summary on the key differences between the

Feature	Wav2Vec	Audio Spectrogram Transformer
Input	Raw audio waveform	Audio spectrogram
Architecture	Transformer with a convolutional front-end	Transformer
Self-supervised learning	Yes	No
Pre-training	On large unlabeled corpora of speech	On large labeled corpora of audio spectrograms
Performance	State-of-the-art on a variety of speech recognition and understanding tasks	State-of-the-art on a variety of audio classification and retrieval tasks

Table 3.1: Comparison of the two foundation models we decided to train using learnable prompts.

two foundation models we decided to employ to test to test how *learnable prompts* work on speech language understanding tasks.

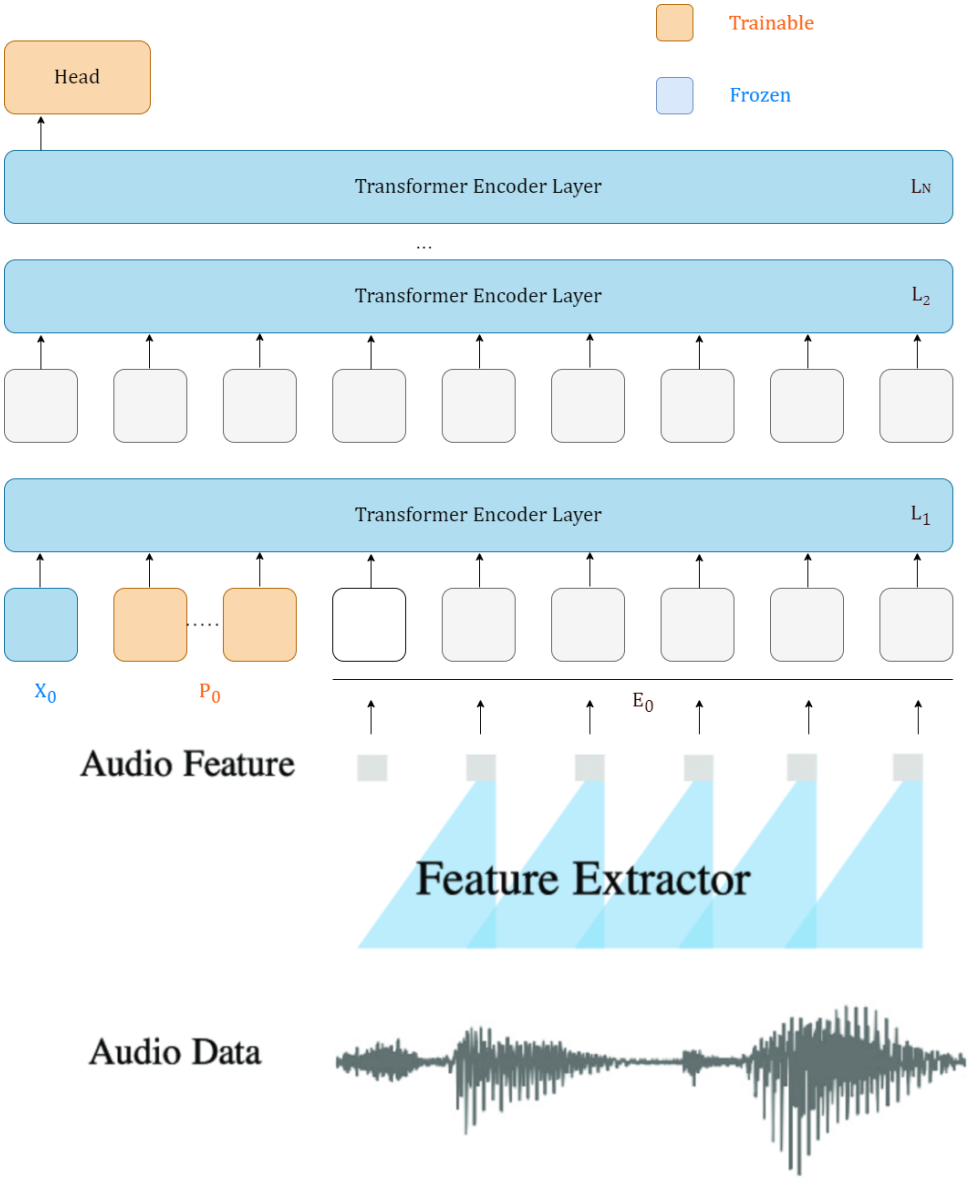

Figure 3.7: SPT, Prompted Architecture Diagram, Wav2Vec

Chapter 4

Dataset and Experiments

In this chapter we'll first focus on defining the task we want to test our models on. Then we'll list some datasets typically used to train SLU models and and go more in detail explaining the dataset we chose to train our models on. Finally, we'll be giving details about the way we implemented the experimental setup and the gathered results which we'll be discussing in the next chapter.

4.1 Dataset and Task

Task Among different SLU tasks, which we briefly introduced in the state-of-the-art chapter, we chose to evaluate the performance of prompting on an E2E SLU model tackling an *intent classification task*. This supervised machine learning task aims to train a model to predict the intent of a spoken utterance. We chose this task because *single-label classification* tasks are generally easier to tackle and implement coding-wise. Additionally, the simplicity of the implementation allowed us to evaluate performance the Wav2Vec 2.0 architecture using the same PyTorch pipeline as the one we built for Audio Spectrogram Transformers (AST), with minimal code adjustments.

Formally, an *intent classification task* is the following: given a set of N labeled examples $D = \{(x_i, y_i)\}_{i=1}^{N}$, where x_i is a speech utterance and y_i is the corresponding intent, the task consists in training a transformer model T to predict the intent of a new speech utterance x:

$$T(x) = \hat{y} \tag{4.1}$$

The goal of the training process is to minimize the cross-entropy loss function[51]:

$$\arg\min_{T} \sum_{i=1}^{N} L(\hat{y}_i, y_i), \tag{4.2}$$

where L is the cross-entropy loss:

$$L(\hat{y}_i, y) = -\sum_{i=1}^{K} y_i \log(\hat{y}_i), \tag{4.3}$$

In the previous equation K is the number of classes and $\hat{y}_i$ can be seen as the softmax probability for the i-th class. $\hat{y}$ is the output of a classification head added to the transformer output.

SLU Datasets Collecting datasets for SLU is more challenging than for ASR as they require not only the transcript but also lables related to the speaker intent. Therefore not many public resources are available. Among them, some datasets worth mentioning are:

- SLURP (Spoken Language Understanding Resource Package)[52]: A large and diverse dataset of spoken language utterances in English, covering a wide range of domains, including travel, weather, music, and navigation. The dataset is labeled with intent, slot values, and dialogue context, making it suitable for a variety of spoken language understanding tasks.

- Spoken-SQuAD [53]: A dataset of spoken question-answer pairs, where the

document is in spoken form and the answer is always a span in the document. The dataset is designed to be challenging for spoken language understanding models, as it requires them to understand both the question and the document in order to generate the correct answer.

- SLUE (Spoken Language Understanding Evaluation) [54]: A suite of benchmark tasks for Spoken Language Understanding Evaluation (SLUE) consisting of limited-size labeled training sets and corresponding evaluation sets. It focuses on ASR and Named Entity Recognition (NER) tasks.

- Fluent Speech Command [55]: A SLU dataset of speech utterances used as commands for smart home devices.

For our experiments we decided to choose to use a more manageable and well-established dataset, which may necessitate the use of an encoder-only Transformer model to address the proposed tasks. Therefore, we decided to work with Fluent Speech Commands, which we'll introduce in the next paragraph.

Dataset The Dataset we choose to test the Prompting approach on is Fluent Speech Command [55], introduced by Fluent.ai, and Mila (Universitè de Montrèal) in 2019. The original paper focuses on Pre-training Techniques for End-to-End Spoken Language Understanding and introduced FSC as a new SLU dataset to show that their method improves performance both when the full dataset or a subset of it is used for training.

The dataset is composed of 16 kHz single-channel .wav audio files. Each audio file contains a recording of a single spoken English smart home/virtual assistant command (e.g. *"Turn on the Bedroom Lights"*). Each audio is labeled with three *slots*: action, object and location, such as *{action: "activate", object: "lights", location:"bedroom"}*. Each slot can take one of multiple values (including "none"). This makes the dataset usable for both:

- *multi-task classification*, defining the task to predict actions, objects and locations.

- *intent classification tasks*, since each combination of slots can be "flattened" and used as 31 distinct labels used for *single-label* classification. For each intent there are multiple possible wordings, for example:

  the intent **{action: "activate", object: "lights", location:"none"}** can be expressed as "turn on the lights", "switch the lights on", "lights on", etc.

Split	# of speakers	# of utterances	# hours
Train	77	23,132	14.7
Valid	10	3,118	1.9
Test	10	3,793	2.4
Total	97	30,043	19.0

Table 4.1: Information about the Fluent Speech Commands dataset (Original Paper [55])

In table 4.1 we can see how the original dataset has been split into Train, validation and test in the original paper. One key point which in this table is the fact that Train, Test and Validation splits are not composed by recordings from the same speakers. This is because researchers want SLU models to be able to generalize and be able to perform Speech Understanding speaker-independently.

Implementation Details: We implemented our models using the *PyTorch* [56] deep learning library, which is known for its flexibility and ease of use. We fine-tuned two pre-trained models from the *Huggingface* Hub: Audio Spectrogram Transformer and Wav2vec2[1]. To gather experimental results we modified the architectures of the downloaded models to adapt them to our proposed approach, converting the models to Torch NN Modules, a standard way of representing neural networks in PyTorch. To do so, we extracted the main architecture components (e.g. projection layers, transformer encoders, etc.) from the predefined classes defined by the original developers and added a prompting mechanism inspired by

[1]We used checkpoints *"MIT/ast-finetuned-audioset-10-10-0.4593"* and *"facebook/wav2vec2-base-960h"*

[40] original code. To do that we defined prompts as learnable parameters and re-defined the NN Module foward function to incorporate prompts alongside the standard input between transformer layers. Additionally, we used *Hydra* [57] to manage our experiments and to easily configure different runs and *Weights&Biases* [58] to log our experiment results and track the performance of our models [2]. We have made all of our code and implementation details available in a public GitHub repository[3] to facilitate the reproducibility of our results and to encourage other researchers to build on our work.

[2]WandB Project Link: https://wandb.ai/sciapponi/AST%20Prompt%20Tuning%20New
[3]Repository Link: *https://github.com/sciapponi/sluprompts*

4.2 Experiments on AST

In this section we'll report the results of our experiments on the Audio Spectrogram Transformer model. We performed tests on both *shallow* adn *deep* prompt tuning techniques and then retrained the models from scratch introducing a learning rate scheduler to see whether it would improve the models' accuracy.

Baselines We defined some baseline results to check the impact of prompting on our proposed architecture accuracy. More specifically:

- **Lower Bound**: defined by a full training on the frozen model. The only weights which are learnt in this experiment consist in the 31 output linear perceptron classification head, connected to the end of the transformer model and used to learn the intents. No prompts have been added for this experiment, which, in our context, is equivalent to performing an experiment with a *prompt length* of 0 tokens. After 50 epochs of training, with batch size = 16 and fixed learning rate = $5 \cdot 10^{-3}$, the model was able to output an intent accuracy of **25.354%** on the *test set*.
- **Upper Bound**: it's defined as the model fine-tuning on the Fluent Speech Command dataset. The model, finetuned for 10 epochs, with batch size = 16 and fixed learning rate = $5e^{-4}$ results in an intent accuracy of **97.30%** on the *test set*.

4.2.1 Shallow Prompt-Tuning SPT

We start our experimental analysis considering shallow prompting were prompt of length N are prepended to the input of the first Transformer Layer. In table 4.2 we can see what percentage of the network parameters is trained during our fine-tuning alternative method. The *Trainable Parameters* column reports the count of all non-frozen parameters of the network, including the classification head. As we can see most experiments trained less than 1% of the network parameters. The

% column contains the percentage of Prompt Parameters with respect to the transformer encoder parameters.

# Tokens	Trainable parameters	Prompt Parameters	%
0	23839	0	0
50	62239	38400	0.045
100	100639	76800	0.090
600	484639	460800	0.54
1600	1252639	1228800	1.44

Table 4.2: SPT, Learnable Parameters

In this table the *0 tokens* example represents the baseline we previously introduced. The only trainable parameters (23,839) consist in the 31 neuron multilayer perceptron added to the end of the transformer to classify the intents. In all other tests the *trainable parameters* of the 0 tokens model are substracted from the *trainable parameters* of the run to compute the *prompt parameters*. These are then used to compute the parameter percentage over the transformer encoder parameters (850M).

# Tokens	Batch Size	Gradient Acc.	Learning Rate	Epochs
0	16	FALSE	5.00E-03	50
50	16	FALSE	5.00E-03	50
100	16	FALSE	5.00E-03	50
600	8	TRUE	5.00E-03	50
1600	4	TRUE	5.00E-03	50

Table 4.3: SPT, Train Hyperparameters

Figure 4.1 presents the plotted results of the experiments. We selected a line plot to visualize the general trend between prompt size and accuracy. We plotted both the intent accuracies for the test and validation sets because the FSC validation set is notoriously difficult and may be of interest as a worst-case scenario for accuracy analysis. All intent accuracies achieved are better than the baseline lower

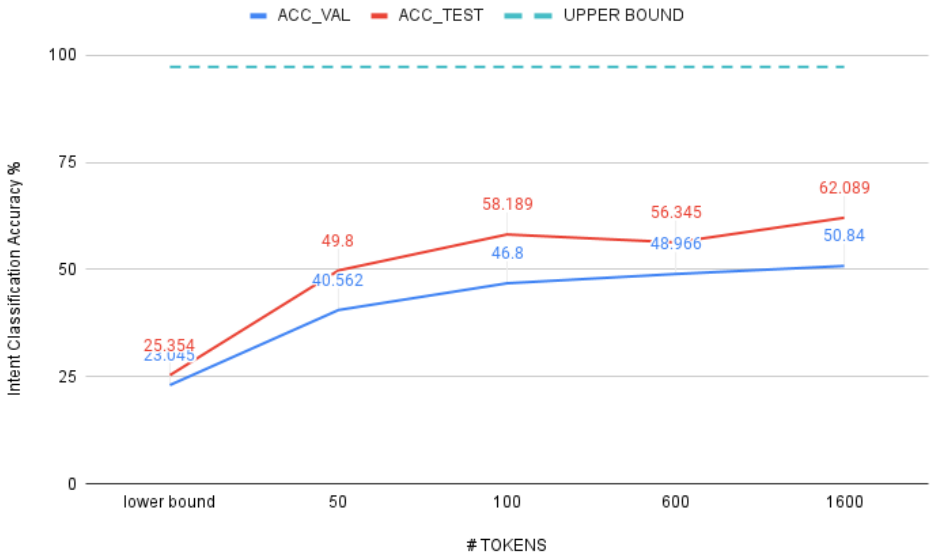

Figure 4.1: SPT, Intent Classification Accuracy on test and validation sets

bound result, but they appear to be significantly lower than the upper bound. Generally, we observe that longer prompts lead to better intent accuracy for the SPT approach. The accuracy improvement does not scale linearly with the number of prompt tokens and appears to saturate around 60% with prompt lengths over 100 tokens.

Figure 4.2 shows a plot of the intent accuracies of the same experiments as figure 4.1, but with an added *cosine annealing learning rate scheduler* [59] at training time. A cosine annealing learning rate scheduler gradually decreases the learning rate during training, following a cosine curve. The learning rate starts at a high value and then gradually decreases to a minimum value, and then increases back to the high value. This process is repeated until the training is complete. This is determined by equation 4.4.

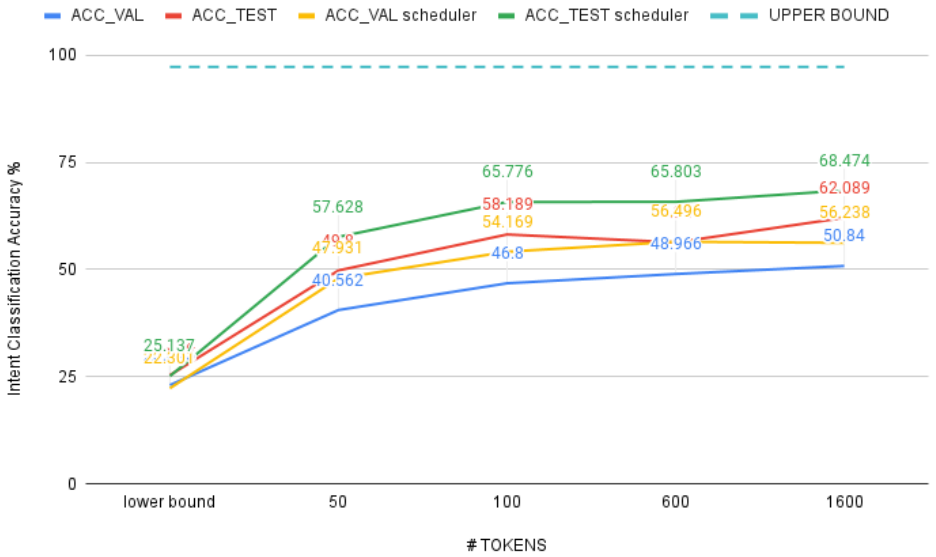

Figure 4.2: SPT vs SPT + scheduler, Intent Classification Accuracy on test and validation sets

$$lr = \eta_{min} + (\eta_{max} - \eta_{min})cos(\pi \frac{t}{T}) \tag{4.4}$$

where:

- η_{min} is the minimum learning rate
- η_{max} is the maximum learning rate
- t is the current training step
- T is the total number of training steps

We decided to apply this type of learning rate scheduler following [40] with $\eta_{max} = 5 \cdot 10^{-3}$, $\eta_{min} = 0$, and $T = 50$. Utilising a cosine annealing learning rate scheduler allowed us to improve the results of about 10% on all the prompt tests, except the base line which doesn't strongly benefit from a lowering learning rate during training. This is most likely because the model was stuck in a local minimum and slowly lowering the learning rate allowed the model to overcome it during the gradient optimization process.

4.2.2 Deep Prompt-Tuning

In this section we'll show training setup and results for experiments with the *DPT* approach, introduced in chapter 3. Tables 4.4 and 4.5 display the percentage of the prompt parameters with respect to the transformer encoder and the training hyperparameters, in a similar fashion as the tables regarding the *SPT* approach.

As we preannounced in the *proposed approach section* the *# Tokens* column in table 4.4 should be multiplied by 12 to calculate the actual size of the prompt parameters learnt by the network. In this context the 50 tokens row is comparable to the *SPT* experiment with 600 tokens, since they both represent 0.54% of the encoder weights.

# Tokens	Trainable parameters	Prompt Parameters	%
50	484639	460800	0.54
100	945439	921600	1.08
200	1867039	1843200	2.17

Table 4.4: DPT, Learnable Parameters

We decided to test three models, doubling the prompt length at each new run, to discover how it would affect the accuracy performances. This resulted in the percentage of parameters also doubling at each new experiment.

# Tokens	Batch Size	Gradient Acc.	Learning Rate	Epochs
50	16	FALSE	5.00E-03	50
100	16	FALSE	5.00E-03	50
200	8	TRUE	5.00E-03	50

Table 4.5: DPT, Training Hyperparameters

In table 4.5 we can see that the hyperparameters are similar to the ones used when testing the SPT approach. The only notable change is the use of gradient accumulation with respect to the overall prompt parameter percentage, since what impacts computation on the DRAM is mostly the input size of the first layer of the transformer, which is handled as input by the CUDA API.

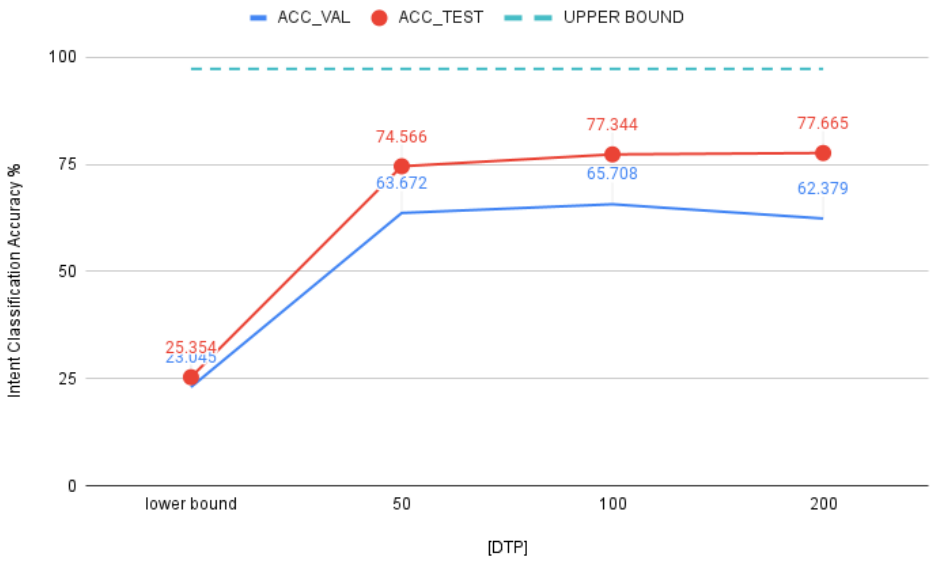

Figure 4.3: DPT, Intent Classification Accuracy on test and validation sets

Figure 4.3, similarly to the result plots for SPT, shows the Intent classification accuracy for each run. Overall, all models were able to achieve a better classification accuracy than SPT, even without employing a learning rate scheduler. Another similar aspect of this run is that the accuracy seems to saturate from a number of prompt tokens onwards, with only slight improvements doubling or quadruplicating the number of prompt weights learnt by the model.

We decided to test the same models with the same hyperparameter configurations, employing the cosine annealing learning rate scheduler. Since it improved the SPT results by around 10%, we expected a similar behaviour in the DPT context. Figure 4.4 shows the Intent Classification accuracy on both *test and validation sets* for the models, on runs with and without scheduler. As we can see from the plot, the employement of a *cosine annealing learning rate scheduler* improves the results between 12 and 15% on both test and validation intent classification accuracy.

Figure 4.5 includes a comparison between the DPT and SPT experiments. The main comparison criteria consists in the percentage of the prompt parameter weights

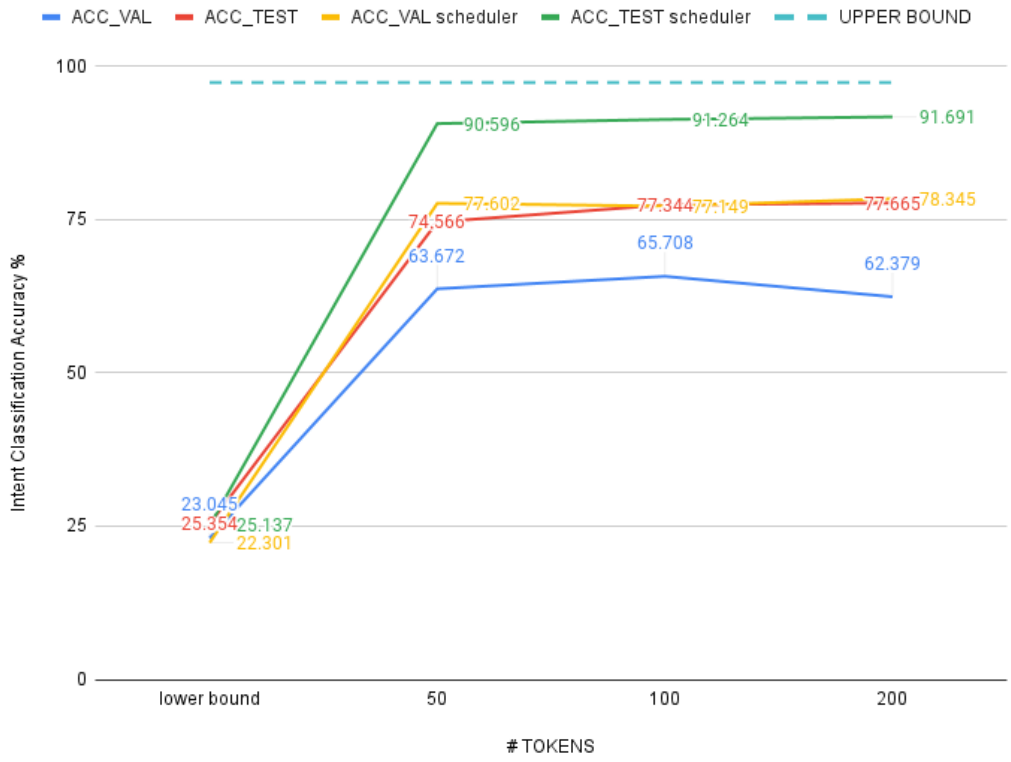

Figure 4.4: DPT + Scheduler, Intent Classification Accuracy on test and validation sets

with respect to the Audio Spectrogram Transformer encoder. The results are composed of the intent accuracy on the test set for various models. To gather comparable results for the 0.043% DPT model we ran an experiment with 4 token prompt length (48 learnt tokens spread throughout the transformer layers). All the experiments we are analysing make use of the *cosine annealing learning rate scheduler*, since using it made us able to achieve better results.

As we can see from the histogram, the DPT approach outperforms SPT on models with a comparable size. The 0.54% model (DPT with 50 tokens) is the best compromise between the highest accuracy with the lowest amount of parameters and outperforms the SPT model of equivalent size (SPT 600 tokens) by about 25% accuracy.

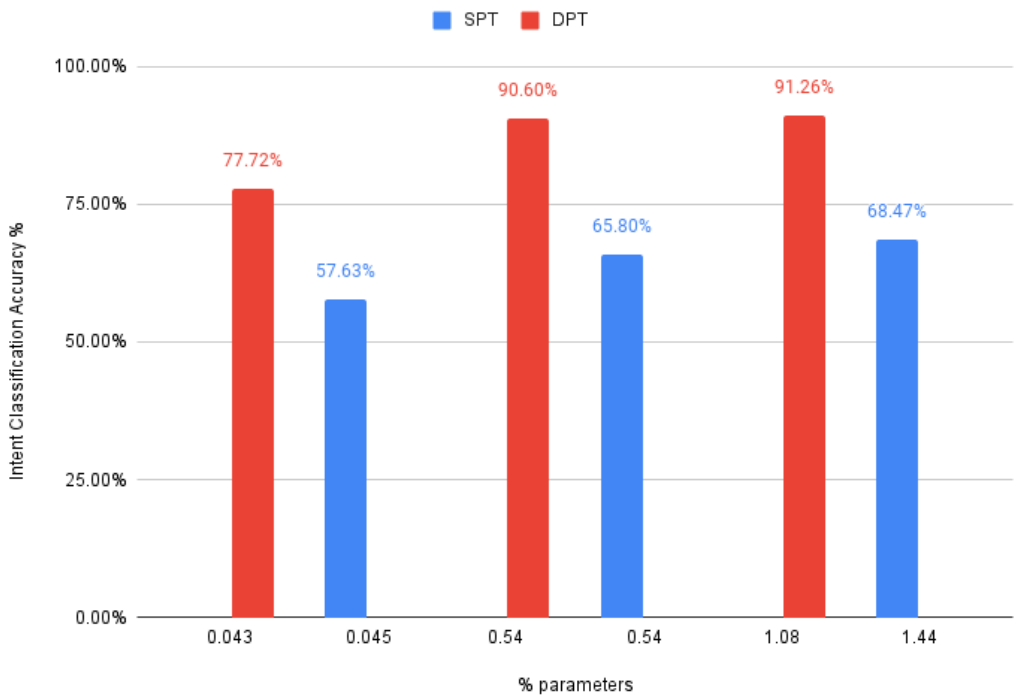

Figure 4.5: Comparison between parameter % comparable DPT and SPT models, intent accuracy on test set

4.3 Tests on Wav2Vec 2.0

To test whether our proposed approach worked on other transformer architectures for SLU we decided to run similar tests for the Wav2Vec2. Because of time constraints we couldn't manage to run extensive experiments on the architecture, meaning we only have some preliminary data. Fortunately results appear to be promising and it could be useful to further investigate them.

Baselines Similarly to what we did in the experiments on Audio Spectrogram Transformers, we defined some baseline results to check the impact of prompting on Wav2Vec2:

- **Lower Bound**: defined by a full training on the frozen model. The only weights which are learnt in this experiment consist in the 31 output linear perceptron classification head, connected to the end of the transformer

model and used to learn the intents. Analogously to the Lower Bound on AST this is equivalent to selecting a *prompt length* of 0 tokens. After 50 epochs of training, with batch size = 16 and fixed learning rate = $5e^{-3}$, the model was able to output an intent accuracy of **81.38%** on the *train set*.

- **Upper Bound**: As we specified in the proposed approach section, Wav2Vec2 achieves an accuracy of **99.7%** on the Fluent Speech Command test set when used for an End-to-End intent Classification. This result was benchmarked by *Seto et Al.* [50]. We have to note that Wav2Vec, diversely from AST is trained on speech data, which means that it genereally works better on SLU tasks.

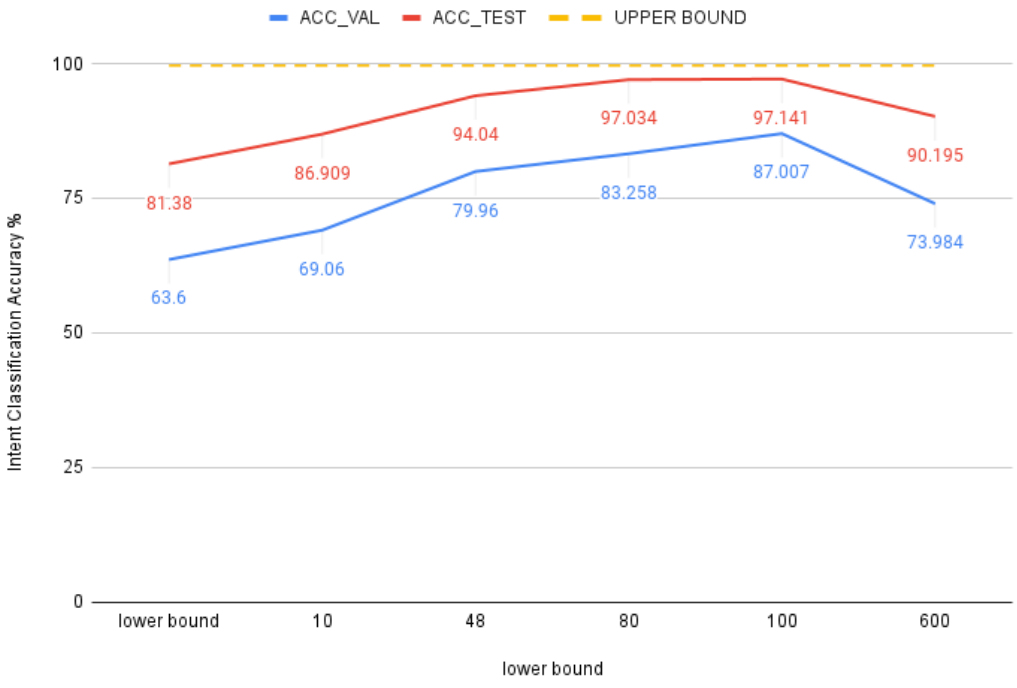

Figure 4.6: Wav2Vec SPT + scheduler, intent accuracy on test and validation sets

Figure 4.6 presents the results of our experiments on Wav2Vec using the SPT prompting technique. All experiments utilized a cosine annealing learning rate scheduler. We chose this type of learning scheduler because it improved the results of our experiments on the Audio Spectrogram Transformer model. All runs

Wav2Vec [SPT]		Wav2Vec [DPT]	
#TOKENS	**%PARAMETERS**	**#TOKENS**	**%PARAMETERS**
10	0.008	7	0.07
48	0.04	50	0.51
80	0.068	100	1.02
100	0.085	200	2.05
600	0.51	300	3.07

Table 4.6: Experiments on Wav2Vec

started with a learning rate of $5 \cdot 10^3$, which was decreased at each epoch using Equation 4.4. We were able to run all experiments with a batch size of 16 because the lack of preprocessing and the use of raw audio as input allowed us to put less pressure on the GPU DRAM.

As shown in the plots, the model achieved optimal results even with a significantly smaller number of parameters than the transformer encoder (see Table 4.3). Overall, the approach was able to classify intents with an accuracy close to the upper bound of 99.7

Figure 4.7 presents the results of our experiments on Wav2Vec using the DPT prompting approach. These results are consistent with our expectations from the AST results, both in terms of intent accuracy on the test and validation sets and in how it relates to the prompt length hyperparameter. Because Wav2Vec does not require preprocessing, training took less time, allowing us to conduct more experimental runs than with Audio Spectrogram Transformers. The 200-token run achieved the best accuracy on both the test and validation sets. Interestingly, a 300-token experiment performed slightly worse than the 200-token experiment, likely due to overfitting, which suggests a limitation of the prompting approach on the Wav2Vec model.

Figure 4.8 compares the performance of the DPT and SPT models on Wav2Vec. The vertical axis represents the model accuracy, and the horizontal axis shows the prompt weights percentage on a log scale. The results show that the SPT model outperforms the DPT model on intent accuracy, even with a significantly smaller

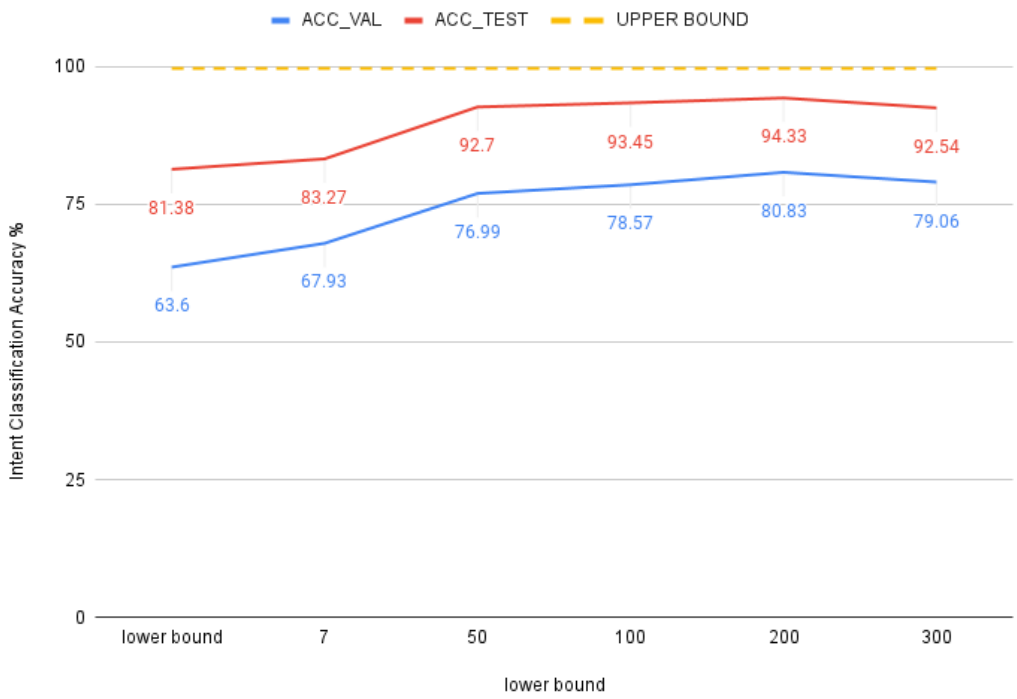

Figure 4.7: Wav2Vec DPT + scheduler, intent accuracy on test and validation sets.

number of parameters. The precise cause of this difference in performance between the two aforementioned methods is still under investigation, but we propose some hypotheses in the next chapter.

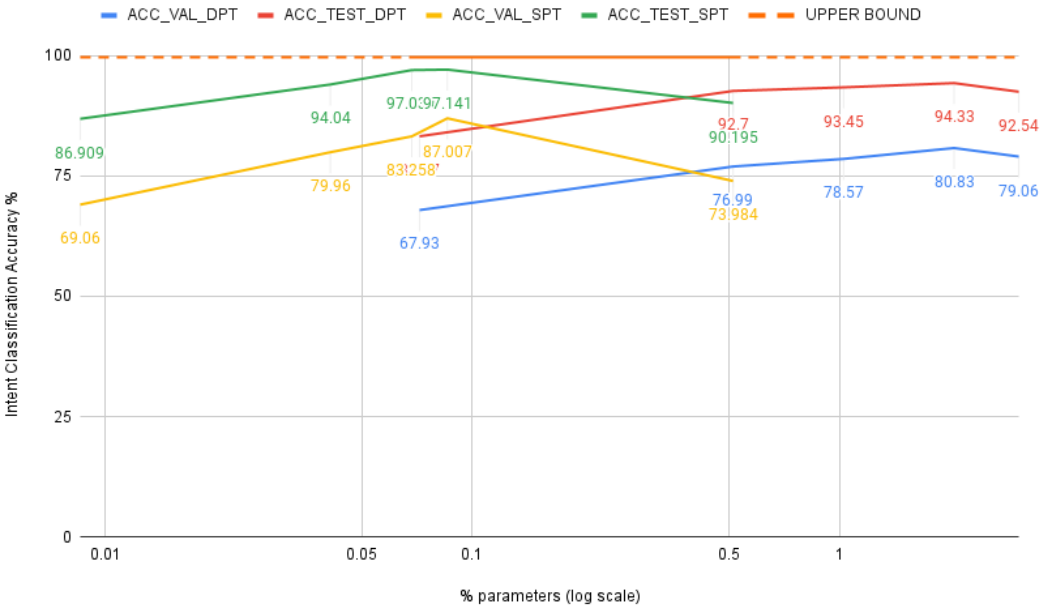

Figure 4.8: Wav2Vec DPT vs SPT, Intent Classification Accuracy on test and validation sets for both approaches. The horizontal axis is on a log scale to make the plot more readable, because of the difference in % of parameters between SPT and DPT.

Chapter 5

Discussion

In this chapter, we will compare and contrast the performance of AST and Wav2Vec on a the experiments results from the previous chapter. We will also focus on Wav2Vec's achievement of 97% intent accuracy on the test set in an experiment with shallow prompt tuning. This result is significant because it suggests that Wav2Vec can be used to achieve close to state-of-the-art results on SLU tasks with our proposed approach. We believe that learnable prompts are a powerful technique that has the potential to revolutionize the way that speech recognition models are deployed and used. In this chapter, we will analyze the results of explore the potential of prompt tuning in more detail.

Model results comparison As demonstrated in the previous chapter, Wav2Vec outperformed AST on the SLU intent classification task. This was anticipated, given that Wav2Vec is pre-trained directly on speech data using contrastive loss to distinguish between real and fake phonemes extracted from raw audio. In contrast, ASTs are trained on sound classification, a more general task that makes the model less specialized and less capable in the SLU domain.

Overall, we demonstrated that learnable prompts are a viable fine-tuning alternative for SLU foundation models. Our results on AST models are inferior to those achieved by Jia et al. [40] (2022) on vision transformers. This is likely due to the inherent complexity of human speech, which is probabilistic in nature from

physical, structural, and semantic perspectives. Also we need to consider that the AST model is, in fact, a Vision Transformer adapted to sound classification tasks using model weights originally trained on image classification tasks. Cross domain transfer learning could lead the model to less general knowledge on speech compared to the ability to classify images of a standard Vision Transformer and achieve worse accuracy. We also have to consider that the input spectrogram size is bigger than the standard 224×224 pixel input size of ViT, making equally long prompt length experiments carry less information in the AST context.

Prompting on Wav2Vec managed to achieve better results than AST. This is most likely due to the different model pretraining and architecture. What's interesting, apart the results themselves, is the way the two different proposed approaches, deep and shallow prompt tuning, have different performances on the two foundation models, namely:

- **AST**: behaved in a similar way as the benchmarks on Vision Transformers by Jia et Al., with better performances on the DPT approach experiments and worse perofrmances on SPT.

- **Wav2Vec**: had better performances using the SPT approach compared to DPT, although DPT still had comparable performances to the ones it had on the AST model. These results are actually closer to the benchmarks from Leister et. Al [38] who worked on Natural Language Processing Tasks using an SPT approach. We have to keep in mind that Leister et Al. tried many different initialization techniques for prompts which we aren't able to translate from a text based to a speech domain, so results are not fully comparable.

Wav2Vec SPT performance The reasons behind Wav2Vec achieving an higher intent classification accuracy than AST with the SPT approach is still unknown to us. This could be attributed to multiple explanations:

- In Wav2Vec's context, prompts are prepended to higher level aggregated speech features (see figure 5.1). This is because we learn and add prompts to the transformer layer *after* the input raw speech is processed through a convolutional backend, which extracts higher level information from it. Prompts representing higher level features, outside the time-frequency realm could lead to more meaningful information needed for a classification task.

- Wav2Vec is more similar in structure to a language model, compared AST. It's trained on a contrastive self-supervised approach, which closely resembles the way text-based language models are trained. This could explain why its results are closer to results from *Leister et Al.* [38] work on language models shallow prompt tuning.

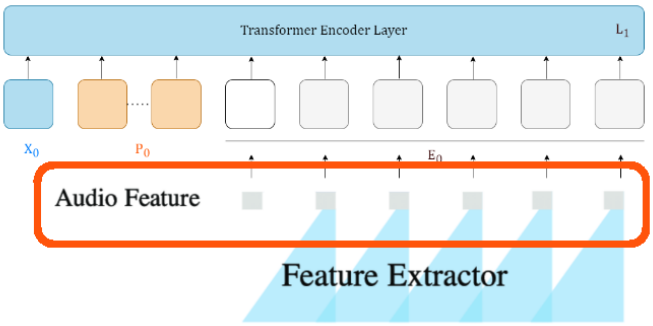

Figure 5.1: Focus on SPT prompting in the Wav2Vec architecture, prompts are prepended to higher level audio features extracted from raw speech.

These explanations are just hypotheses and should be further investigated to have a proper explanation to why Wav2Vec works this well with a *shallow prompt tuning* approach.

Chapter 6

Conclusion and Future Improvements

In this thesis, we explored the use of learnable prompts as a fine-tuning alternative on transformer-based models for Speech Language Understanding. We specifically tested shallow (SPT) and deep (DPT) prompt tuning techniques on two different models: Audio Spectrogram Transformer (AST) and Wav2Vec 2.0. We evaluated the performance of these approaches on an intent classification task, which involved predicting the intent of speech commands for smart home devices.

Our results demonstrated that learnable prompts can be used as a viable alternative to fine-tuning for specific application domains. Some of our experiments achieved intent classification accuracies close to those of models fine-tuned with a standard approach, which modifies all the model's pre-trained weights. However, we were able to achieve this performance by keeping the model frozen and learning new weights corresponding to around 1% or less of the models' encoder.

We believe that our work has several important implications. First, it demonstrates that learnable prompts can be used to effectively fine-tune transformer-based models for SLU tasks. Second, our results suggest that learnable prompts can be used to fine-tune SLU models without requiring significant changes to the model's architecture or pre-trained weights. This is important because it can reduce the computational cost and complexity of fine-tuning SLU models.

In the following paragraphs, we discuss some potential avenues for future research that could extend and complement the work presented in this thesis.

Further tests and experiments To further confirm our hypothesis, we could run experiments and tests with different directions.

- Test both SPT and DPT on different SLU tasks, such as named entity recognition, dialogue state tracking, or speech information extraction. This would allow us to assess the generalizability of our findings to other SLU tasks.
- Run experiments on more models specifically pre-trained for SLU, to see whether their experimental results would come closer to Wav2Vec, favoring SPT to DPT. This would help us to understand whether the superior performance of Wav2Vec on SPT is due to the fact that it is pre-trained on a large corpus of speech data.
- Since the *Leister et al.* paper states that the larger the model, the closer prompt-tuning results resemble standard fine-tuning, it would be interesting to test prompt tuning on larger models. This would help us to understand the relationship between the size of the model and the effectiveness of prompt tuning in a SLU domain.
- Experiment on prompt transferability, both cross-model and cross-task. Prompt transferability [42] refers to the ability to use a prompt that has been trained on one model or task to improve the performance of another model or task. This is an area that is worth researching for its utility in many use cases, such as when we have limited labeled data, when we want to quickly adapt a model to a new task or when we want to transfer knowledge to a different SLU model.

In addition to these experiments, we could also explore other avenues for improving the performance of prompt tuning for SLU tasks. For example, we could investigate new techniques for designing and training learnable prompts, or we could

explore the use of prompt tuning in conjunction with other machine learning techniques, such as meta-learning [60] or transfer learning [61].

Possible Case Studies One potentially fruitful avenue for evaluating the capabilities of our approach would be through case-study research. This would allow us to leverage the small size of learnable prompts to solve practical problems where model size is a critical consideration.

For example, we could use learnable prompts trained on different tasks to implement a multi-task SLU API. In such a system, a frozen model would leverage different prompts, each specialized in a different task, depending on the type of inference request being received. This approach would have several advantages over traditional SLU systems, which typically is based on specialized model each one trained to respond to each task. First, it would allow us to develop a more compact and efficient SLU model. Second, it would enable us to fine-tune the performance of individual tasks by training specialized prompts. Finally, it would make it easier to add new tasks to the system without having to retrain the entire model.

Another aspect which would be interesting to investigate is the use of prompting in a federated learning [43] setting. Federated learning is a privacy-preserving machine learning technique that allows devices to collaboratively train a model without sharing their data. Each device trains the model on its own data and sends the updated model parameters to a central server. The server aggregates the updates from all devices and uses them to update the global model, which is then sent back to the devices. In a federated learning setting prompts could be useful to reduce communication overhead. This is because communicating the model updates to the central server can be a significant burden on communication bandwidth and battery life. Learnable prompts can reduce this overhead by allowing clients to communicate only a small number of parameters, rather than the entire model. Moreover, federated learning is often used in settings where the data on

different clients is heterogeneous, meaning that it comes from different distributions. Learnable prompts can help to address this challenge by allowing clients to learn prompts that are specific to their own data distribution. This can make the global model more robust to data heterogeneity and improve its performance on all clients. Finally learnable prompts could be used for Rehearsal-free federated continual learning: the task of t raining a model to learn new tasks without forgetting the knowledge it has learned from previous tasks. Learnable prompts can be used to facilitate knowledge transfer between tasks and clients in rehearsal-free federated continual learning in a similar way as [41].

Bibliography

[1] L. E. Baum and T. Petrie, "Statistical inference for probabilistic functions of finite state markov chains," *The annals of mathematical statistics*, vol. 37, no. 6, pp. 1554–1563, 1966.

[2] L. Bahl, P. Brown, P. de Souza, and R. Mercer, "Maximum mutual information estimation of hidden markov model parameters for speech recognition," in *ICASSP '86. IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 11, 1986, pp. 49–52. DOI: 10.1109/ICASSP.1986.1169179.

[3] B. H. Juang and L. R. Rabiner, "Hidden markov models for speech recognition," *Technometrics*, vol. 33, no. 3, pp. 251–272, 1991. DOI: 10.1080/00401706.1991.10484833. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/00401706.1991.10484833.

[4] M. Gales and S. Young, "The application of hidden markov models in speech recognition," *Foundations and Trends® in Signal Processing*, vol. 1, no. 3, pp. 195–304, 2008, ISSN: 1932-8346. DOI: 10.1561/2000000004. [Online]. Available: http://dx.doi.org/10.1561/2000000004.

[5] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.

[6] A.-r. Mohamed, G. E. Dahl, and G. Hinton, "Acoustic modeling using deep belief networks," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 1, pp. 14–22, 2012. DOI: 10.1109/TASL.2011.2109382.

[7] O. Abdel-Hamid, A.-r. Mohamed, H. Jiang, and G. Penn, "Applying convolutional neural networks concepts to hybrid nn-hmm model for speech recognition," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2012, pp. 4277–4280. DOI: 10.1109/ICASSP.2012.6288864.

[8] D. Wang, X. Wang, and S. Lv, "An overview of end-to-end automatic speech recognition," *Symmetry*, vol. 11, no. 8, 2019, ISSN: 2073-8994. DOI: 10.3390/sym11081018. [Online]. Available: https://www.mdpi.com/2073-8994/11/8/1018.

[9] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd International Conference on Machine Learning*, ser. ICML '06, Pittsburgh, Pennsylvania, USA: Association for Computing Machinery, 2006, pp. 369–376, ISBN: 1595933832. DOI: 10.1145/1143844.1143891. [Online]. Available: https://doi.org/10.1145/1143844.1143891.

[10] R. L. Stratonovich, "Conditional markov processes," in *Non-linear transformations of stochastic processes*, Elsevier, 1965, pp. 427–453.

[11] D. Zhu, H. Yao, B. Jiang, and P. Yu, *Negative log likelihood ratio loss for deep neural network classification*, 2018. arXiv: 1804.10690 [cs.LG].

[12] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *Proceedings of the 31st International Conference on Machine Learning*, E. P. Xing and T. Jebara, Eds., ser. Proceedings of Machine Learning Research, vol. 32, Bejing, China: PMLR, 22–24 Jun 2014, pp. 1764–1772. [Online]. Available: https://proceedings.mlr.press/v32/graves14.html.

[13] A. Hannun, "Sequence modeling with ctc," *Distill*, 2017, https://distill.pub/2017/ctc. DOI: 10.23915/distill.00008.

[14] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. eprint: https://direct.mit.edu/neco/article-pdf/9/8/1735/813796/neco.1997.9.8.1735.pdf. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735.

[15] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional lstm networks," in *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, vol. 4, 2005, 2047–2052 vol. 4. DOI: 10.1109/IJCNN.2005.1556215.

[16] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017. arXiv: 1706.03762. [Online]. Available: http://arxiv.org/abs/1706.03762.

[17] M. F. Stollenga, J. Masci, F. Gomez, and J. Schmidhuber, "Deep networks with internal selective attention through feedback connections," in *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, Eds., vol. 27, Curran Associates, Inc., 2014. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2014/file/19de10adbaa1b2ee13f77f679fa1483a-Paper.pdf.

[18] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, pp. 107–116, Apr. 1998. DOI: 10.1142/S0218488598000094.

[19] S. Karita, N. Chen, T. Hayashi, *et al.*, "A comparative study on transformer vs RNN in speech applications," *CoRR*, vol. abs/1909.06317, 2019. arXiv: 1909.06317. [Online]. Available: http://arxiv.org/abs/1909.06317.

[20] D. Povey, A. Ghoshal, G. Boulianne, *et al.*, "The kaldi speech recognition toolkit," in *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*, IEEE Catalog No.: CFP11SRW-USB, Hilton Waikoloa Village, Big Island, Hawaii, US: IEEE Signal Processing Society, Dec. 2011.

[21] C.-X. Qin, W.-L. Zhang, and D. Qu, "A new joint ctc-attention-based speech recognition model with multi-level multi-head attention," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2019, no. 1, p. 18, Oct. 2019, ISSN: 1687-4722. DOI: 10.1186/s13636-019-0161-0. [Online]. Available: https://doi.org/10.1186/s13636-019-0161-0.

[22] W. Lin, "Drawbacks of transformers," Apr. 2023.

[23] V. Pratap, A. Sriram, P. Tomasello, *et al.*, *Massively multilingual asr: 50 languages, 1 model, 1 billion parameters*, 2020. arXiv: 2007.03001 [eess.AS].

[24] J. Bai, B. Li, Y. Zhang, *et al.*, "Joint unsupervised and supervised training for multilingual asr," in *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 6402–6406. DOI: 10.1109/ICASSP43922.2022.9746038.

[25] A. Baevski and A. Mohamed, "Effectiveness of self-supervised pre-training for asr," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 7694–7698. DOI: 10.1109/ICASSP40776.2020.9054224.

[26] X. Zheng, Y. Liu, D. Gunceler, and D. Willett, "Using synthetic audio to improve the recognition of out-of-vocabulary words in end-to-end asr systems," in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 5674–5678. DOI: 10.1109/ICASSP39728.2021.9414778.

[27] J. Pfeiffer, A. Rücklé, C. Poth, *et al.*, "Adapterhub: A framework for adapting transformers," *CoRR*, vol. abs/2007.07779, 2020. arXiv: 2007.07779. [Online]. Available: https://arxiv.org/abs/2007.07779.

[28] N. Houlsby, A. Giurgiu, S. Jastrzebski, *et al.*, "Parameter-efficient transfer learning for NLP," in *Proceedings of the 36th International Conference on Machine Learning*, K. Chaudhuri and R. Salakhutdinov, Eds., ser. Proceedings of Machine Learning Research, vol. 97, PMLR, Sep. 2019, pp. 2790–2799. [Online]. Available: https://proceedings.mlr.press/v97/houlsby19a.html.

[29] N. Houlsby, A. Giurgiu, S. Jastrzebski, *et al.*, *Parameter-efficient transfer learning for nlp*, 2019. arXiv: 1902.00751 [cs.LG].

[30] B. X. B. Yu, J. Chang, L. Liu, Q. Tian, and C. W. Chen, *Towards a unified view on visual parameter-efficient transfer learning*, 2023. arXiv: 2210.00788 [cs.CV].

[31] D. Yin, Y. Yang, Z. Wang, H. Yu, K. Wei, and X. Sun, "1% vs 100%: Parameter-efficient low rank adapter for dense predictions," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2023, pp. 20 116–20 126.

[32] H. Chen, R. Tao, H. Zhang, *et al.*, *Conv-adapter: Exploring parameter efficient transfer learning for convnets*, 2022. arXiv: 2208.07463 [cs.CV].

[33] H. Le, J. Pino, C. Wang, J. Gu, D. Schwab, and L. Besacier, *Lightweight adapter tuning for multilingual speech translation*, 2021. arXiv: 2106.01463 [cs.CL].

[34] X. Gong, Y. Lu, Z. Zhou, and Y. Qian, "Layer-wise fast adaptation for end-to-end multi-accent speech recognition," in *Interspeech 2021*, ISCA, Aug. 2021. DOI: 10.21437/interspeech.2021-1075. [Online]. Available: https://doi.org/10.21437%2Finterspeech.2021-1075.

[35] B. Zhang, X. Jin, W. Gong, *et al.*, *Multimodal video adapter for parameter efficient video text retrieval*, 2023. arXiv: 2301.07868 [cs.CV].

[36] Z. Jiang, F. F. Xu, J. Araki, and G. Neubig, "How Can We Know What Language Models Know?" *Transactions of the Association for Computational Linguistics*, vol. 8, pp. 423–438, Jul. 2020, ISSN: 2307-387X. DOI: 10.1162/tacl_a_00324. eprint: https://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00324/1923867/tacl_a_00324.pdf. [Online]. Available: https://doi.org/10.1162/tacl%5C_a%5C_00324.

[37] Y. Su, X. Wang, Y. Qin, *et al.*, "On transferability of prompt tuning for natural language processing," in *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Association for Computational Linguistics, 2022. DOI: 10.18653/v1/2022.naacl-main.290. [Online]. Available: https://doi.org/10.18653%2Fv1%2F2022.naacl-main.290.

[38] B. Lester, R. Al-Rfou, and N. Constant, "The power of scale for parameter-efficient prompt tuning," *CoRR*, vol. abs/2104.08691, 2021. arXiv: 2104.08691. [Online]. Available: https://arxiv.org/abs/2104.08691.

[39] Z. Yang, S. Wang, B. P. S. Rawat, A. Mitra, and H. Yu, *Knowledge injected prompt based fine-tuning for multi-label few-shot icd coding*, 2022. arXiv: 2210.03304 [cs.CL].

[40] M. Jia, L. Tang, B.-C. Chen, *et al.*, "Visual prompt tuning," in *European Conference on Computer Vision*, Springer, 2022, pp. 709–727.

[41] Z. Wang, Z. Zhang, C. Lee, *et al.*, "Learning to prompt for continual learning," *CoRR*, vol. abs/2112.08654, 2021. arXiv: 2112.08654. [Online]. Available: https://arxiv.org/abs/2112.08654.

[42] N. F. Liu, M. Gardner, Y. Belinkov, M. E. Peters, and N. A. Smith, "Linguistic knowledge and transferability of contextual representations," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*

(Long and Short Papers), Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 1073–1094. DOI: 10.18653/v1/N19-1112. [Online]. Available: https://aclanthology.org/N19-1112.

[43] H. B. McMahan, E. Moore, D. Ramage, and B. A. y Arcas, "Federated learning of deep networks using model averaging," *CoRR*, vol. abs/1602.05629, 2016. arXiv: 1602.05629. [Online]. Available: http://arxiv.org/abs/1602.05629.

[44] A. Dosovitskiy, L. Beyer, A. Kolesnikov, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *CoRR*, 2020. arXiv: 2010.11929. [Online]. Available: https://arxiv.org/abs/2010.11929.

[45] Y. Gong, Y. Chung, and J. R. Glass, "AST: audio spectrogram transformer," *CoRR*, vol. abs/2104.01778, 2021. arXiv: 2104.01778. [Online]. Available: https://arxiv.org/abs/2104.01778.

[46] Y. Muthusamy, R. Cole, and M. Slaney, "Speaker-independent vowel recognition: Spectrograms versus cochleagrams," in *International Conference on Acoustics, Speech, and Signal Processing*, 1990, pp. 533–536. DOI: 10.1109/ICASSP.1990.115767.

[47] P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," *CoRR*, vol. abs/1804.03209, 2018. arXiv: 1804.03209. [Online]. Available: http://arxiv.org/abs/1804.03209.

[48] A. Baevski, H. Zhou, A. Mohamed, and M. Auli, *Wav2vec 2.0: A framework for self-supervised learning of speech representations*, 2020. arXiv: 2006.11477 [cs.CL].

[49] K. Choi and E. J. Yeo, *Opening the black box of wav2vec feature encoder*, 2022. arXiv: 2210.15386 [cs.SD].

[50] S. Seo, D. Kwak, and B. Lee, *Integration of pre-trained networks with continuous token interface for end-to-end spoken language understanding*, 2022. arXiv: 2104.07253 [cs.CL].

[51] I. J. Good, "Rational decisions," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 14, no. 1, pp. 107–114, 1952, ISSN: 00359246. [Online]. Available: http://www.jstor.org/stable/2984087 (visited on 09/21/2023).

[52] E. Bastianelli, A. Vanzo, P. Swietojanski, and V. Rieser, *Slurp: A spoken language understanding resource package*, 2020. arXiv: 2011.13205 [cs.CL].

[53] C.-H. Li, S.-L. Wu, C.-L. Liu, and H.-y. Lee, *Spoken squad: A study of mitigating the impact of speech recognition errors on listening comprehension*, 2018. arXiv: 1804.00320 [cs.CL].

[54] S. Shon, A. Pasad, F. Wu, *et al.*, *Slue: New benchmark tasks for spoken language understanding evaluation on natural speech*, 2022. arXiv: 2111.10367 [cs.CL].

[55] L. Lugosch, M. Ravanelli, P. Ignoto, V. S. Tomar, and Y. Bengio, *Speech model pre-training for end-to-end spoken language understanding*, 2019. arXiv: 1904.03670 [eess.AS].

[56] A. Paszke, S. Gross, F. Massa, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[57] O. Yadan, *Hydra - a framework for elegantly configuring complex applications*, Github, 2019. [Online]. Available: https://github.com/facebookresearch/hydra.

[58] L. Biewald, *Experiment tracking with weights and biases*, Software available from wandb.com, 2020. [Online]. Available: `https://www.wandb.com/`.

[59] I. Loshchilov and F. Hutter, *Sgdr: Stochastic gradient descent with warm restarts*, 2017. arXiv: `1608.03983 [cs.LG]`.

[60] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey, *Meta-learning in neural networks: A survey*, 2020. arXiv: `2004.05439 [cs.LG]`.

[61] F. Zhuang, Z. Qi, K. Duan, *et al.*, *A comprehensive survey on transfer learning*, 2020. arXiv: `1911.02685 [cs.LG]`.

Acknowledgements

I'm very grateful to Alessio Brutti and Umberto Cappellazzo for guiding me through my internship at Fondazione Bruno Kessler. I'd also like to acknowledge my advisor, prof. Paolo Torroni, for sending some timely advice when needed.