POLITECNICO DI TORINO

Master's Degree in Computer Engineering

Master's Degree Thesis

Speech-Text Cross-Modal Learning through Self-Attention Mechanisms

Supervisors

Eliana PASTOR

Co-Supervisors

Alkis KOUDOUNAS

Moreno LA QUATRA

Candidate

Damiano BONACCORSI

December 2023

Abstract

Speech, with its various elements like intonation and non-verbal vocalizations, is considered to be the earliest form of human language. However, existing systems for understanding spoken language mostly focus on the textual aspect, disregarding these additional components. Recent advancements in speech language modeling have enabled the development of speech-based language models called SpeechLMs. Nevertheless, text remains the primary mode of communication on the internet. Given this pretext, the objective of the thesis is to analyze the current state-of-the-art speech models, and design a novel approach to combine the speech and text modalities, to obtain an architecture that is capable of leveraging the advantages of both. To do so, we decided to base our approach around the ideas of two previous works named respectively VisualBERT [1] and AST [2].

In VisualBERT, the authors introduce a versatile framework for various vision-and-language tasks by leveraging BERT[3], a well-known transformer-based text model. This approach combines regions of an input image (as "visual tokens") with textual captions, feeding them into the Transformer[4] layers that then align the visual and textual tokens using self-attention.

In AST (Audio Spectrogram Transformer), the authors introduce the first entirely convolution-free, transformer-based model for audio classification, moving beyond traditional approaches that combined convolutional neural networks (CNNs) with self-attention mechanisms. The authors suggest that solely relying on attention[4] mechanisms could be sufficient for audio classification tasks.

In our work, we aim to extend and adapt the innovative approach of VisualBERT to encompass the speech-text domain. This adaptation involves a key modification: instead of using visual tokens derived from regions of an input image, we propose to use patches extracted from a 2D audio spectrogram. These spectrogram patches are obtained by following the preprocessing methodology outlined in AST for an input speech sample.

We name our model SpectroBERT, and like its predecessor did for the image-text modalities, we believe SpectroBERT will be able to implicitly align text and speech features, while retaining the simple and flexible formulation originally proposed in VisualBERT. Our approach also aims to explore how the self-attention mechanism can be utilized to bridge the gap between auditory and textual information.

Acknowledgements

I want to begin these acknowledgments with heartfelt gratitude to my family, as without their unwavering support I would not be here celebrating such an achievement. In particular, a special mention goes to my mother, that in spite of the adversities life presented her with, has raised me and my brother to become the individuals we are today, and enabled us each to pursue our studies in our respective paths of interest, doing her best to always give us her support.

My appreciation extends to all the people that have stood by me during these formative years. To Michelangelo, with whom I shared years of lectures, projects, laughs and coffees - I thank you - may our startup come soon. To Loris and Edoardo, 'de bois', for the countless hours spent gaming and chilling on Discord - I thank you - may our yearly dose of Minecraft keep our paths together.

To my Vulcanus family, special thanks go to Stefano, Isabel, Carlos, and Cindy, the #AtsuGang, for being the best neighbours I could have asked for; and to Athena, for her words of love and comfort received when I needed them the most - a warm thank you.

Lastly, I express my profound gratitude to my thesis' supervisor Eliana and co-supervisors Alkis and Moreno. Your guidance through the development of this thesis has been invaluable, and I want to thank you for having provided me the opportunity to delve into a project that was both challenging yet interesting, and from which I gained insights and knowledge that I will treasure in my future professional life.

Table of Contents

List of Tables

List of Figures

Acronyms

AI

Artificial Intelligence

AQA

Audio Question Answering

ASR

Automatic Speech Recognition

AUROC

Area Under the Receiver Operating Characteristic Curve

BERT

Bidirectional Encoder Representations from Transformers

CLS

Classification

CNN

Convolutional Neural Network

DL

Deep Learning

LSTM

Long Short Term Memory

ML

Machine Learning

MLM

Masked Language Modeling

MLP

Multilayer Perceptron

NLP

Natural Language Processing

NLU

Natural Language Understanding

RCNN

Region Convolutional Neural Network

RNN

Recurrent Neural Network

ROC

Receiver Operating Characteristic

Seq2Seq

Sequence to Sequence

SER

Speech Emotion Recognition

SOTA

State Of The Art

Chapter 1

Introduction

The internet's evolving communication landscape has undergone a transformation in recent years, with a growing shift towards multimodal interaction, blending text with other forms of communication like images, videos, and voice messages.

While written text remains the dominant mode of online communication, the emergence of Machine Learning speech-based language models, is making spoken language and non-verbal elements more integral to the digital experience. These models enable greater accessibility and inclusivity, benefiting individuals with visual impairments, literacy challenges, or those who simply prefer spoken communication.

However, understanding spoken language presents unique challenges related to accents, background noise, and regional dialects, which ongoing advancements aim to address. Moreover, human-computer interaction increasingly integrates speech technology, creating more seamless interactions through virtual assistants, chatbots, and voice-controlled devices. In a globally connected internet, recognizing cultural and linguistic diversity is essential, and advances in natural language processing are working towards bridging the gap between text and speech understanding, making online communication more comprehensive and nuanced.

1.1 Thesis objective

Given this context, the central objective of this thesis is to propose a novel approach to seamlessly integrate the spoken and textual domains, while giving a comprehensive overview of the previous works that led to its formulation. We aim to construct a framework that leverages the strength of both modalities, and to do so we draw inspiration from two works specifically: VisualBERT [1] and AST [2].

Our approach will be explained extensively in the following chapters, but a brief summary for the inpatients is provided here.

Starting from VisualBERT [1]'s framework, we adapt it to bridge the realms of speech and text - rather than images and text - by replacing the visual component (the image encoder) with the audio preprocessing method introduced in AST [2], as per Figure 1.1. The end result is a model that *sees* audio spectrograms rather than images, hence the choice for its name: *SpectroBERT*.

There are other small changes and adjustments that need to be made in order to enable SpectroBERT to work properly, but those will be discussed later, together with the finer details that make the transition from images to spectrograms possible.

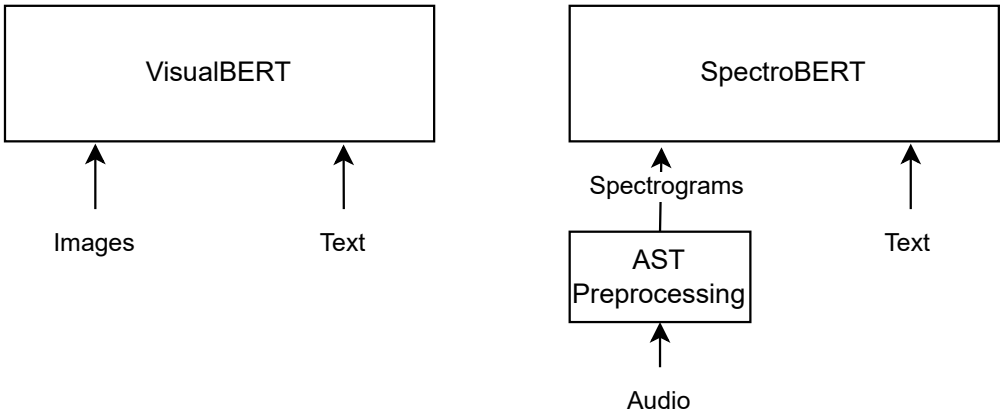

Figure 1.1: Simplified scheme of our suggested approach. Here, the visual component is replaced to accomodate spectrograms instead.

1.2 Background

Before delving into the technical details of SpectroBERT, we find it appropriate to provide a brief background of the concepts that are integral to this work, and also introduce the terminology that is used throughout it, so that even an inexperienced reader (in the field) can familiarise with those terms and ideas that are common when dealing with modern Artificial Intelligence, and understand what is being treated.

In introducing the concepts, we try following a (chrono-)logical order, taking into account the first time they appeared on a publication and highlighting relations between them when possible, but we also include Figure 1.2 as a "map" to further assist the reader in orienting themselves within the various mentioned notions.

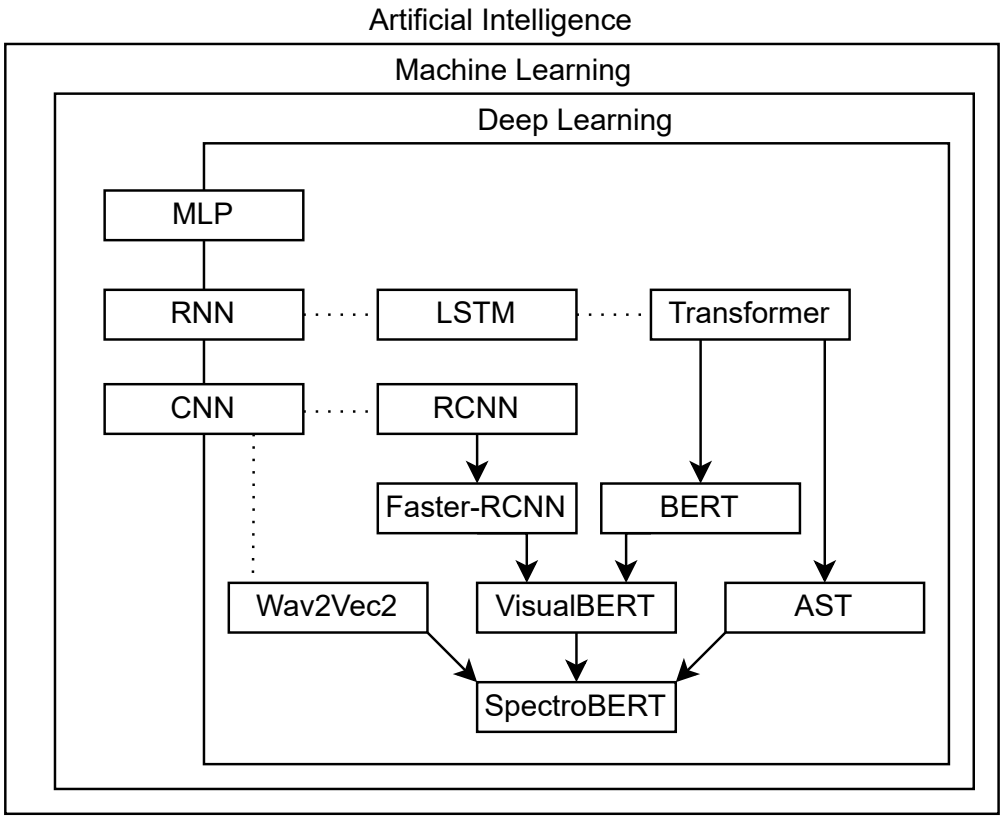

Figure 1.2: Concepts' map. Here we highlight the relations between the concepts and previous works that are integral to our work. This map is not an exact reference as including all connections would be too messy, so some notations are not reported for the sake of clarity. For example, Wav2Vec2 also has a Transformer component, but we do not use it in our work, so that connection has been removed.

1.2.1 Artificial Intelligence

The growth in popularity that Artificial Intelligence has seen in recent years might trick any person into believing that it is a relatively new technology. However, the origin of the idea of a machine capable of simulating human-like intelligence has a fascinating history that can be traced back far long before modern electronic computers were even a thing, as shown in the following by some interesting anecdotes.

Ancient Myths and Legends. Even in ancient civilizations, there were stories that hinted at the idea of artificial intelligence, like in Greek mythology for example, where Hephaestus, the god of craftsmanship, was believed to have created mechanical servants. These myths allude to the notion of fabricated intelligence

that predates modern computing technology by millennia.

The Mechanical Turk. In the 18th century, there was a famous automaton known as the "Turk", an elaborate mechanical contraption that appeared to play chess. In truth, the machine had a human operator hidden inside, but it still sparked public interest and discussion about the possibility of creating machines that could one day mimic human intelligence.

Ada Lovelace's Insights. Ada Lovelace, a visionary figure in the 19th century, worked on the design of the Analytical Engine: a mechanical, general-purpose computer. Her remarkable insights and notes suggested the potential for machines to perform tasks beyond mere number crunching, foreshadowing modern general-purpose computation and the possibility of machine intelligence.

Alan Turing's Turing Machine. In the 1930s, mathematician and logician Alan Turing introduced the concept of the "Turing machine" [5], a theoretical model of computation that laid the foundation for modern computing and had a profound influence on the development of Artificial Intelligence. His insights into the nature of computation and algorithms were pivotal in shaping the field.

So, while the term "Artificial Intelligence" was formally introduced at the Dartmouth Conference in 1956, the idea of creating intelligent machines or simulating human-like intelligence has been a topic of interest and speculation for centuries, and the fact that AI has reached mainstream adoption in recent years only, is merely a consequence of the exponential growth that computing capabilities underwent during the last decade.

Progress in computer chip manufacturing, with increasingly more computational power at our disposal, is what enabled the AI field to blossom and establish itself as a technological revolution, and while the term encompasses a variety of techniques and approaches, it is nowadays most closely associated with one in particular: Machine Learning.

1.2.2 Machine Learning

Machine Learning, a subset of Artificial Intelligence, proposes a shift in the programming paradigm such that instead of using rule-based systems relying heavily on the *explicit* programming of rules, we make use of data-driven systems relying on algorithms that can learn *implicit* patterns and rules directly from data, allowing for more flexibility and adaptability.

In other words, where programmers previously had to write each and every rule/condition by hand to obtain a desirable output, Machine Learning allows instead to automatically infer those rules by setting up a learning scheme between the desirable output itself and the data, as illustrated in figure 1.5.

Figure 1.3: AI through AI's eyes. This collection of images has been generated using Microsoft's Bing AI Image Generator [6], with a simple "Artificial Intelligence" textual prompt. It's interesting to see how AI imagines itself.

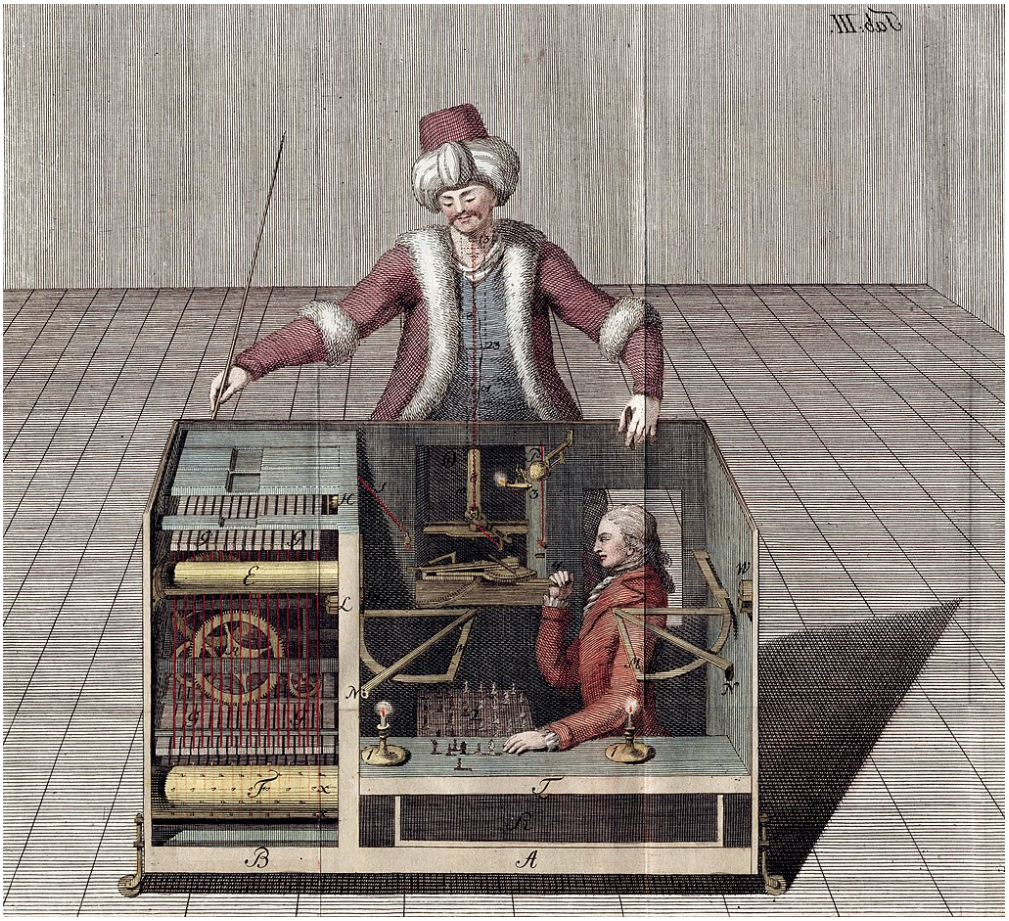

Figure 1.4: The Mechanical Turk. This cross-section drawn by Joseph von Racknitz (1744–1818), shows how he thought the operator sat inside, controlling the fake Turk as it was facing the human opponent. This image is considered public domain.

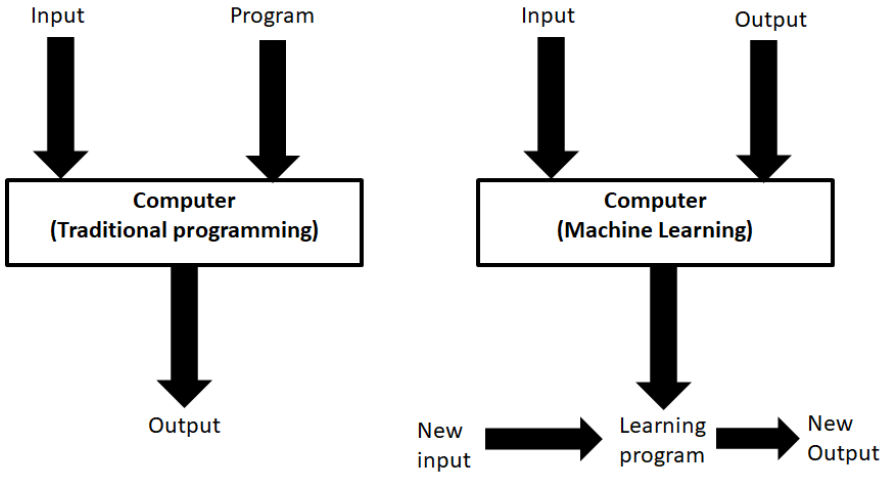

Figure 1.5: Machine Learning's paradigm shift. Compared to traditional programming, we feed the ML algorithm with inputs and desired outputs, and the model learns the rules that connect the two automatically.

There are many ML techniques, and covering all of them is out of the scope of this work, but most of them (with exceptions) usually follow this simplified learning scheme:

1. **Collect, clean, and prepare data** of interest, or use an existing dataset if possible, and preprocess them to be fed into the ML model, eventually removing samples that could harm the learning process, like outliers.

2. **Select relevant features** from data, this is usually done by means of an Exploratory Data Analysis, where features that are not discerning enough between samples are discarded.

3. **Select a suitable model** depending on factors like the "nature" of the data found in the previous step, the task at hand, the computational resources at disposal. Models can have many formulations, but in general they all reduce to a parametric mathematical function:

$$y = f(x, m) \quad \text{where } f \text{ is the chosen model and } m \text{ are its parameters}$$

4. **Infer the outputs** by feeding the data into the model:

$$\hat{y}_i = f(x_i, m) \quad \text{for each sample } x_i \text{ in the dataset}$$

5. **Compute loss** by comparing the obtained outputs to the desirable ones, using a *loss function* to quantify the error (usually a distance) between them. A simple formulation follows:

$$Loss = \hat{y} - y = \sum_i (\hat{y}_i - y_i)$$

where y_i is the desirable output (or *ground truth*) for sample x_i

6. **Adjust the model** by tweaking its parameters, depending on the loss.
7. **Repeat** steps 4-6 until the gradually-tweaked parameters make the model perform as close as possible to the desired behavior, i.e. the loss value has been minimized (or maximized, depending on its formulation).

Once the ML model has been trained, it can be used to make predictions on unseen data, as in Figure 1.6.

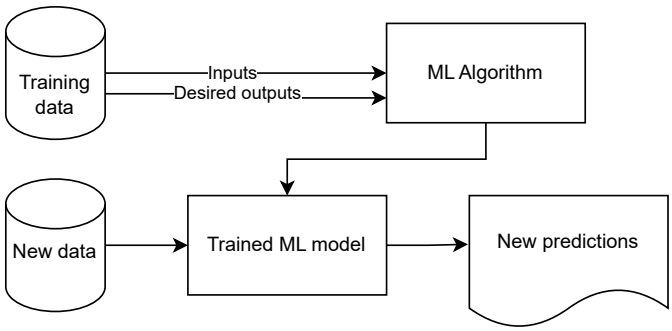

Figure 1.6: Machine Learning scheme.

1.2.3 Deep Learning

Deep Learning is an advanced form of Machine Learning that has affirmed itself as the pinnacle of this discipline. The idea is to employ neural networks with many multiple successive layers - hence the term *deep* - to model high-level abstractions in data, enabling machines to make decisions with minimal human intervention, and often surpassing human performance in tasks like image and speech understanding.

Deep Learning architectures learn feature hierarchies leveraging large amounts of data, by feeding this data into the succession of layers, and then having each layer

in the network transform its inputs into a slightly more abstract and composite output representation, that becomes the input for the next layer. In an image processing context for example, the hierarchy might consist of edges in the early layers, combinations of edges in the middle layers, and complete object parts in deeper layers.

This ability to perform hierarchical feature extraction is where the strength of Deep Learning lies. Contrary to traditional ML techniques that require manual intervention to select and extract the features, Deep Learning allows for fully-automatic feature extraction from raw data.

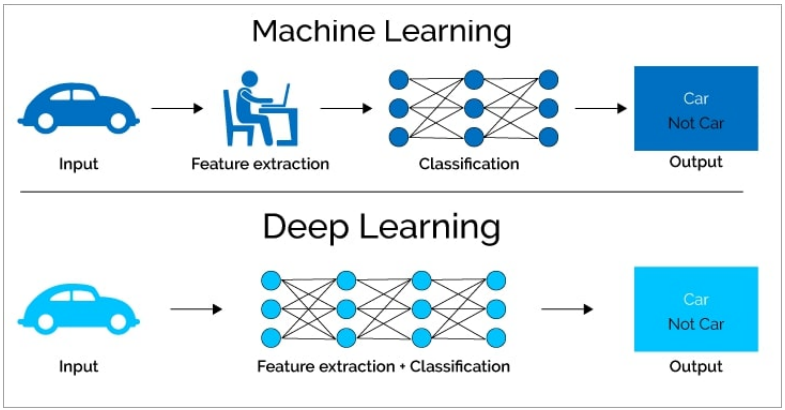

Figure 1.7: Deep Learning versus Machine Learning. In DL the manual component is replaced by deeper networks capable of performing feature extraction automatically. Credits to the original author [7].

The essential unit of these networks is the artificial neuron, which loosely models the neurons in a biological brain. Just like a biological neuron receives input from its dendrites and produces output through its axon, the artificial neuron in a neural network applies a weighted sum to its input, adds a bias, and then passes it through a non-linear activation function, as in Figure 1.8.

Mathematically, the process within a single neuron can be expressed as:

$$output = f_{activation}\left(\sum_i w_i input_i + bias\right) \tag{1.1}$$

Table 1.1 showcases the mathematical formulation of some of the most commonly-used activation functions.

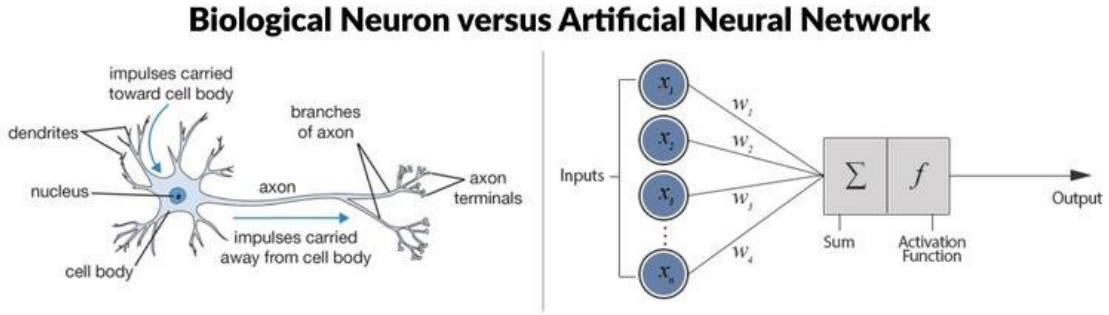

Figure 1.8: Biological Neuron versus Artificial Neuron. Credits to the original author [8].

ReLU	$f(x) = \begin{cases} 0 & \text{for } x \leq 0 \\ x & \text{for } x > 0 \end{cases}$
Softmax	$f_i(\vec{x}) = \frac{e^{x_i}}{\sum_{j=1}^{J} e^{x_j}} i = 1, ..., J$
tanh	$f(x) = \tanh(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})}$

Table 1.1: Examples of activation functions, which can operate either element-wise or vector-wise.

Deep Learning's success is largely attributed to the ever-growing availability of massive datasets and substantial computational power. As previously mentioned, ML models are usually trained by minimizing a loss function that measures the difference between the predicted and true values, so the combination of automatic feature extraction with the increase in data and computational resources has directly translated to more sophisticated and accurate models. Moreover, Neural Networks have not limited themselves to linear combinations of artificial neurons, but different ways to interconnect neurons and transform data within each layer have been proposed, as explained in the following sections.

1.2.4 CNN

Convolutional Neural Networks (CNNs) have revolutionized the domain of image analysis, harnessing the power of Deep Learning to interpret visual data with unprecedented accuracy and efficiency.

At the heart of a CNN is the convolutional layer, which applies a series of small learnable filters by sliding them over the entire input image - hence the name *convolutional*, inspired by the mathematical convolution - obtaining feature maps that capture the input image's salient features and patterns such as edges, textures, and shapes.

After the convolutional layer, usually follows the pooling layer - often referred to as a downsampling layer - that serves to reduce the spatial dimensions of the feature maps, which not only lowers the computational load for subsequent layers, but also embeds a form of translation invariance into the network, which means that patterns in the image can be recognized irrespective of their location in the visual field.

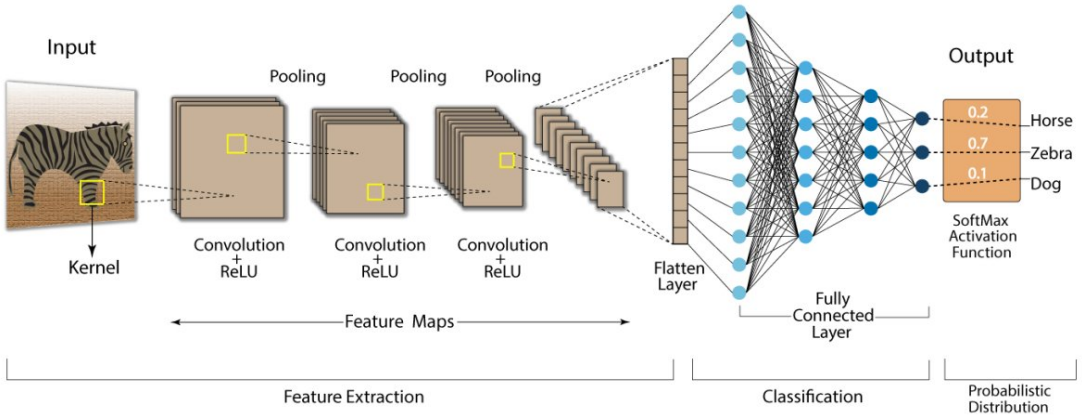

Figure 1.9: Example of Convolutional Neural Network. Credits to the original author [9].

As the image data traverses through successive convolutional and pooling layers, the CNN abstracts higher-level features, transitioning from generic to specific representations. This hierarchical feature extraction is similar to the way the human visual cortex interprets visual stimuli, layering simple and complex patterns to construct a comprehensive understanding of the scene. The CNN's architecture is not limited to images though, but can handle any kind of grid-like 2-dimensional data.

1.2.5 R-CNN and Faster-RCNN

Region Convolutional Neural Network [10] (R-CNN, not to be confused with Recursive Neural Networks) pioneered a revolutionary approach to visual tasks, transforming the way objects are identified and localized in images. Differently from conventional sliding window methods, which proved way too computationally intensive as the image resolution increased, R-CNN introduced the innovative

concept of region proposals, which are the only regions of the image which will be processed while all the other regions are ignored.

R-CNN initiates the process by generating these region proposals through a method known as selective search, which involves over-segmenting the image into small regions that are progressively merged based on similarity criteria. The outcome is a set of region proposals spanning multiple scales and aspect ratios, and each undergoes processing by a Convolutional Neural Network to extract a fixed-length feature vector.

How these feature vectors are used is totally up to the researcher and depends on the downstream task, but in the case of Object Detection like in the original paper, the feature vectors extracted by the CNN serve a dual purpose, addressing both classification and bounding box regression.

For classification, a Support Vector Machine [11] (SVM) is employed to categorize objects within each region proposal into specific classes or as background. Meanwhile, bounding box regression is managed by a linear regressor, which infers the coordinates of each bounding box, i.e. the smallest rectangle containing the entirety of the object. The final step in R-CNN is a technique known as non-maximum suppression (NMS), which prunes multiple detections of the same object by discarding bounding boxes with too high of an overlap ratio.

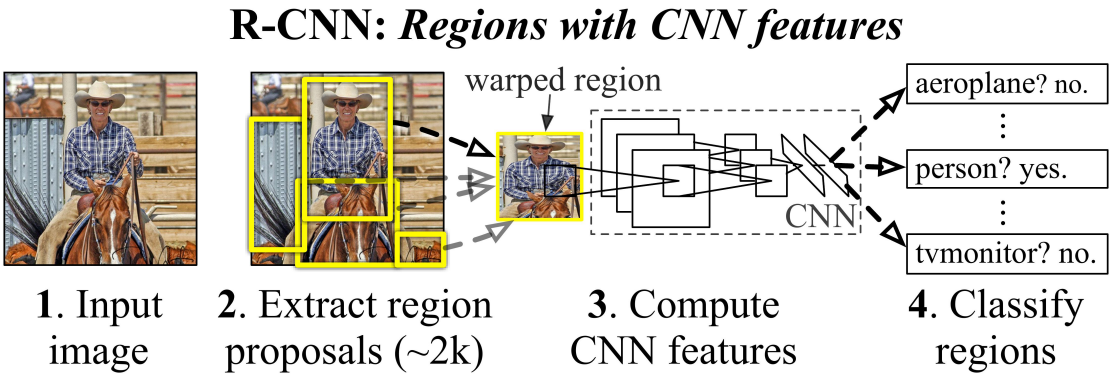

Figure 1.10: R-CNN. Picture from the original paper [10].

While R-CNN [10] marked a transformative breakthrough in object detection, its computational demands led to its evolution into Faster R-CNN [12]. This subsequent model introduced optimizations to enhance computational efficiency and detection speed, incorporating a Region Proposal Network (RPN) with the CNN model. Unlike R-CNN, Faster R-CNN shares full-image convolutional features with the detection network, thereby significantly reducing computational overhead associated with proposal generation.

Faster R-CNN operates as an end-to-end deep convolutional network, leveraging the Region Proposal Network for efficient region proposal generation. The architecture of Faster R-CNN involves a two-stage process. First, the RPN generates region proposals — potential locations of objects. Each proposal then undergoes CNN processing to extract a fixed-length feature vector. This vector serves a dual purpose: classification using a softmax layer to determine the object class, and bounding box regression to refine localization. By integrating the RPN, Faster R-CNN achieves improved efficiency without compromising the quality of region proposals.

Faster R-CNN's innovative approach addressed the computational bottlenecks of its predecessors, providing a more holistic and efficient solution for object detection tasks. The integration of RPN exemplifies a strategic optimization, allowing Faster R-CNN to outperform R-CNN in terms of both speed and accuracy. This model stands as a testament to the iterative nature of advancements in computer vision, showcasing how refined architectures can strike a balance between computational efficiency and detection precision in complex object detection applications.

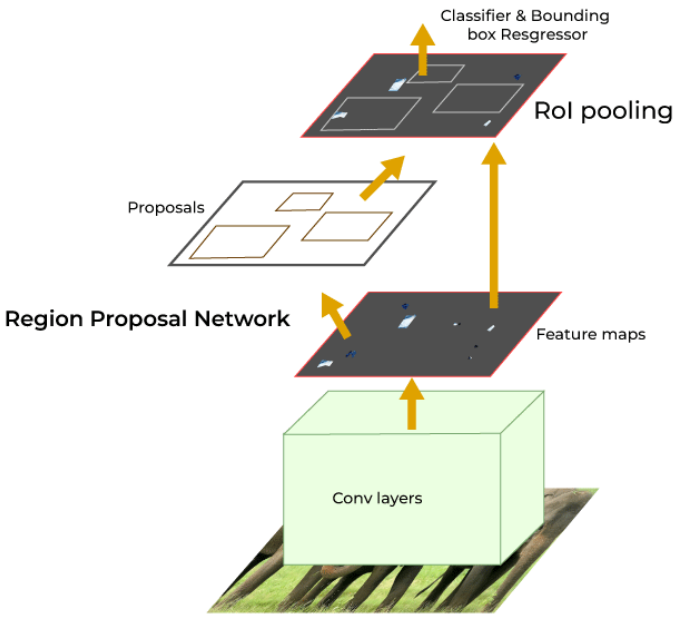

Figure 1.11: Faster R-CNN. Picture from the original paper [12].

1.2.6 RNN and LSTM

Recurrent Neural Networks [13] (RNNs) and Long Short-Term Memory [14] (LSTM) networks are considered pivotal architectures in the field, as they were specifically designed to handle sequences of data rather than a fixed-length input.

RNNs [13] work by incorporating feedback loops in the network architecture, which allow information to persist over time, enabling the network to perform tasks that involve temporal dependencies such as language modeling and time-series prediction. However, RNNs often struggle with long-term dependencies due to issues like vanishing or exploding gradients, where the information from earlier inputs becomes lost or excessively magnified as it propagates through the network.

LSTMs [14] try to address these shortcomings by introducing a more complex unit within the RNN architecture. An LSTM unit includes several *gates* that regulate the flow of information: the input gate, forget gate, and output gate. The input gate controls the extent to which a new value flows into the cell, the forget gate determines what details are discarded from the cell state, and the output gate decides what part of the cell state makes it to the output. These gates work in unison to control the cell state, allowing LSTMs to *remember* important information over long periods and forget the non-essential details.

Mathematically, the operations within an LSTM unit can be represented as follows:

$$
\begin{aligned}
i_t &= \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}) \\
f_t &= \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) \\
g_t &= \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}) \\
o_t &= \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \\
c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\
h_t &= o_t \odot \tanh(c_t)
\end{aligned}
$$

Here

- x_t represents the input at time step t
- h_t is the hidden state at time step t
- c_t is the cell state at time step t
- i_t, f_t, and o_t are the input, forget, and output gates' activations, respectively
- g_t is the cell input activation vector

- W terms denote weight matrices, b terms denote bias vectors,
- σ represents the sigmoid function and $\odot$ denotes the element-wise multiplication.

LSTMs have been a significant step in advancing the state of the art in a multitude of domains that require dealing with sequential patterns, ranging from speech recognition to text generation, thanks to their ability to deal with long-range temporal dependencies.

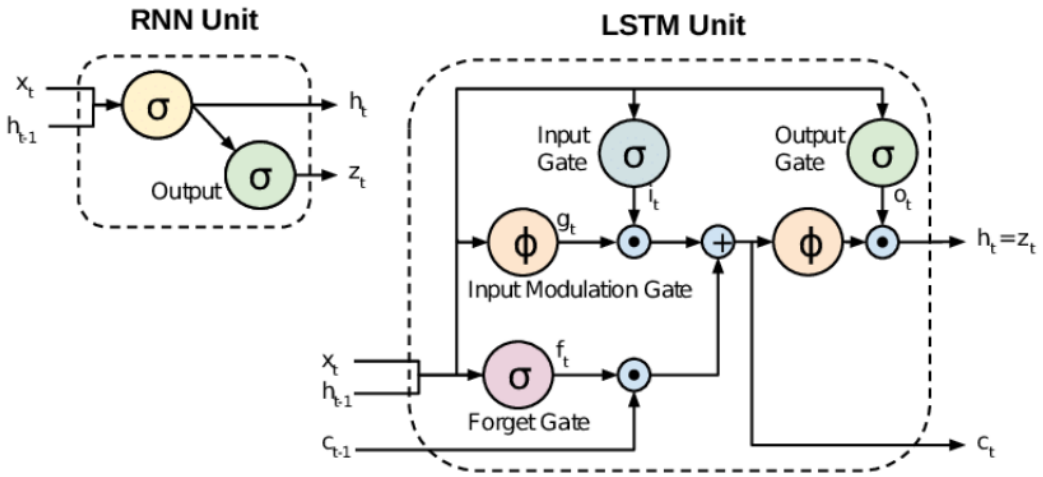

Figure 1.12: RNN and LSTM. Credits to the original author [15].

1.2.7 Transformer

The Transformer [4] architecture too revolutionized the field by introducing a novel approach to sequence-to-sequence tasks. Unlike RNNs, which handle data sequentially by processing one sample of the sequence at a time, Transformers handle sequences in parallel, allowing for greater efficiency and speed.

The architecture of a Transformer consists of two main components: an encoder and a decoder. Each of these components is composed of multiple identical layers (six in the original paper), and each layer within these components has two sublayers: a multi-head self-attention mechanism and a position-wise fully connected feed-forward network.

The multi-head self-attention mechanism is the heart of the Transformer model, because it's what enables the model to focus on different parts of the input sequence when making predictions. This mechanism works by computing a weighted sum of

input features, where the weights are dynamically determined based on the input data themselves - hence the name *self*. Mathematically:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}$$

where $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ represent the Query, Key, and Value matrices, respectively, and d_k is the dimensionality of the key vectors. In other words, the attention is a continuous value between 0 and 1 that weights the *importance* of each sample in relation to the others.

The encoder takes an input sequence and maps it to a sequence of continuous representations, which aim to capture the context of each item in the sequence with respect to all the other items in the sequence.

The decoder, on the other hand, generates an output sequence from these representations, making use of self-attention to focus on the important places in the input sequence.

In traditional RNNs, information from earlier time steps gets diluted over time, making it hard for the model to handle long sequences, but thanks to the attention mechanism Transformers overcome this limitation, allowing for direct connections between all pairs of samples in the input sequence, which explains their ability to handle long-range dependencies.

Being an integral part to this work, more details on Transformers and the attention mechanism will be provided in the later chapters, where these concepts are explained in the context of our work.

1.2.8 BERT

Bidirectional Encoder Representations from Transformers [3], or BERT, is a Transformer-based model that revolutionized the field of Natural Language Processing (NLP) since its introduction by Google AI Language in 2018.

Historically, traditional models would look at text sequentially, either from left to right or right to left, but BERT's core idea is to make the most of the attention mechanism introduced with Transformers, which allows it to *'read'* text in both direction at once - hence the term *bidirectional*. This simultaneous consideration of context from both directions is what makes BERT exceptionally good at understanding the nuances and complexity of language.

An important concept to introduce now is BERT's training strategy, as many aspects of our own training approach are derived from it.

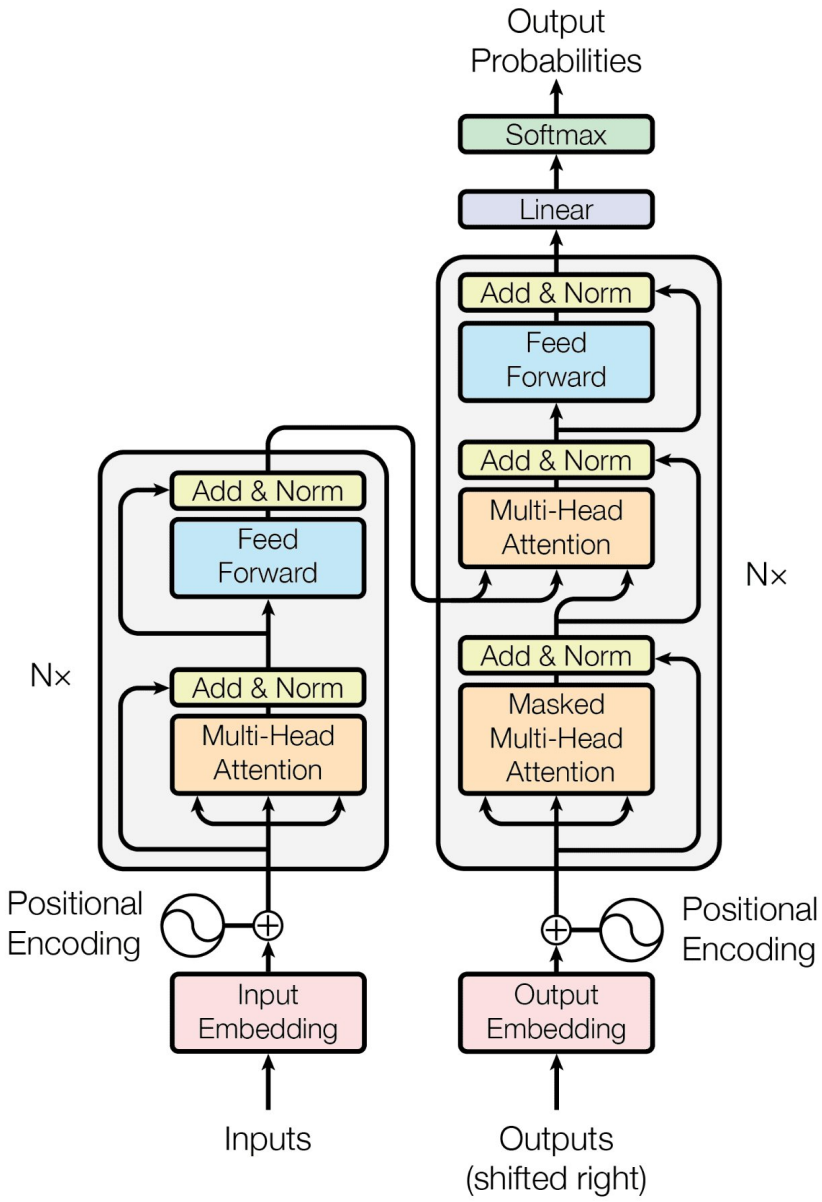

Figure 1.13: Transformer architecture. Picture from the original paper [4].

First, BERT is pre-trained with a composition of two objectives: a Masked Language Modeling (MLM) one, and a Next Sentence Prediction (NSP) one. This means that during pre-training a given percentage of tokens is masked and the model must predict their original value - the MLM objective - while also predicting if two sentences are consecutive or not - the NSP objective.

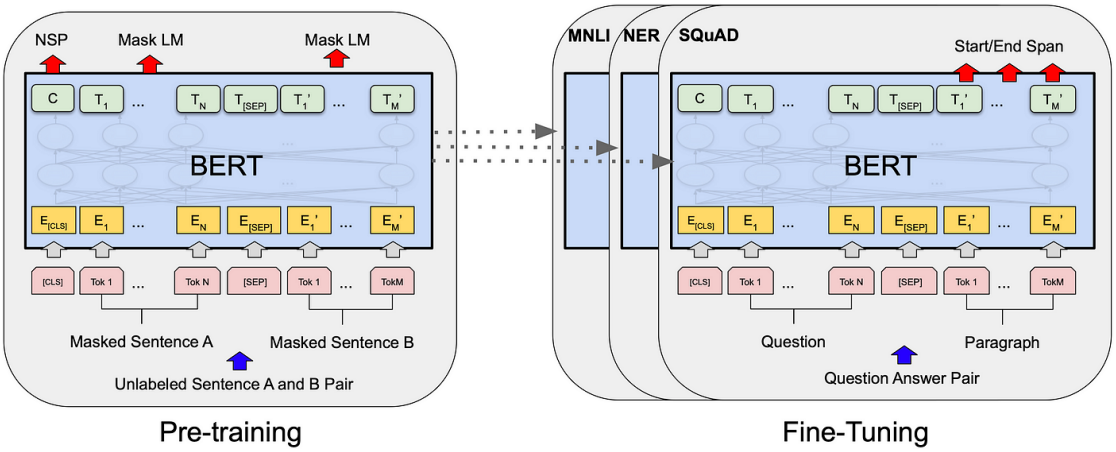

Figure 1.14: BERT architecture. Picture from the original paper [3].

After pre-training, with only an additional output layer, BERT can easily be fine-tuned to excel in a wide array of different NLP tasks, from text classification to question answering, without extensive modifications to its architecture. This ability to transfer its pre-trained knowledge to specific tasks with minimal fine-tuning has made it a popular choice for researchers and practitioners looking for a robust, versatile, and relatively easy-to-use solution in the field of AI.

1.2.9 Cross-Modal Learning

Cross-Modal Learning, often referred to as Multimodal Learning, is an increasingly important concept in the Artificial Intelligence and Machine Learning field. As the name suggests, it involves training systems to understand and integrate information from various sources of different modality, like images, sounds, and text. This approach is key to creating models that can make sense of the complex and varied data we encounter in real-world situations.

At its heart, cross-modal learning is about teaching models to find correlations between different types of data. For example, in speech-text models like ours, the system learns how spoken words correspond to their written form. This isn't just about being able to process each type of data separately; it's about merging these different data sources into a shared unified understanding, by filling in the gaps between what is seen, heard, and read, trying to make the most out of the unique strengths of each data type to improve its overall performance.

However, developing cross-modal learning models isn't straightforward. It

require architectures that can handle complex, high-dimensional data and find meaningful connections across different data types, and also demands large datasets with diverse, multimodal content. The training process must also be adapted to ensure the model not only understands each type of data on its own, but also learns how these different data types interact and complement one another. In essence, cross-modal learning aims to create systems that are more intuitive and intelligent, by leveraging multiple forms of data like the intricate human perception works.

1.3 Related works

1.3.1 VisualBERT

VisualBERT [1] has emerged as a significant model in the landscape of multimodal learning, specifically in tasks that require the understanding of visual content alongside textual information. It is a model that extends the capabilities of BERT to the realm of vision and language tasks, enabling the processing of a combination of image regions and textual descriptions through the self-attention mechanism of the Transformer architecture.

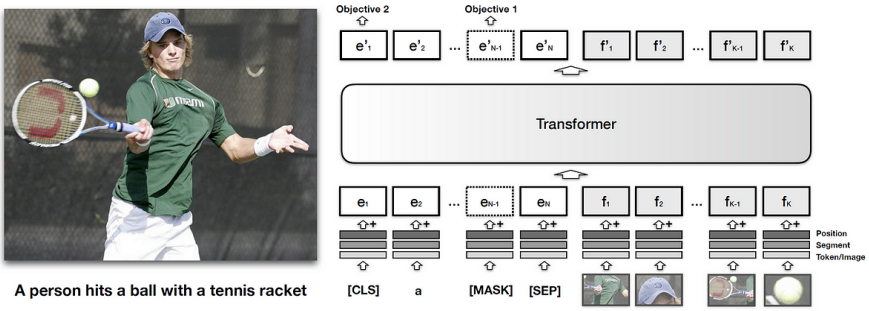

Figure 1.15: VisualBERT. Picture from the original paper [1].

In the development of SpectroBERT, we took a lot of inspiration from VisualBERT's structure. Essentially, we're using its clever way of handling different types of information and tweaking it to work with audio and text instead of images and text. We adops VisualBERT's strategy of combining tokens from two distinct data sources - audio and text in our case - into one stream that's then processed by a Transformer [4] network. The goal here is to understand how spoken words and their written forms are related, helping us get a better grasp of how different types of data interact with each other.

What sets our SpectroBERT apart is how we have adapted VisualBERT's approach. Instead of using visual tokens derived from image regions, we introduce 'audio tokens' extracted from spectrogram representations of audio data. This shift lets us use the self-attention mechanism of the Transformer to explore how audio is related to text, similar to how VisualBERT correlates visual and textual data.

1.3.2 AST

The Audio Spectrogram Transformer [2] (AST) is a cutting-edge tool in audio processing that uses a Transformer [4] model to analyze and classify audio data. A key part of AST is how it prepares audio for analysis: it turns audio signals into two-dimensional spectrograms, which allows the model to interpret audio like visual data, using techniques from image recognition tasks which are usually the domain of Convolutional Neural Networks (CNNs).

When building SpectroBERT, we borrowed this idea from AST. We convert audio into spectrogram patches, which is like turning sounds into many visual snippets, that capture both the temporal and frequency details of speech, crucial for understanding it properly. This method lets us feed 'audio images' into our model, similar to how VisualBERT handles visual data. The great thing about using these spectrogram patches is that it helps our model's attention mechanism focus on specific parts of the audio, just like when we humans focus on certain parts of a picture.

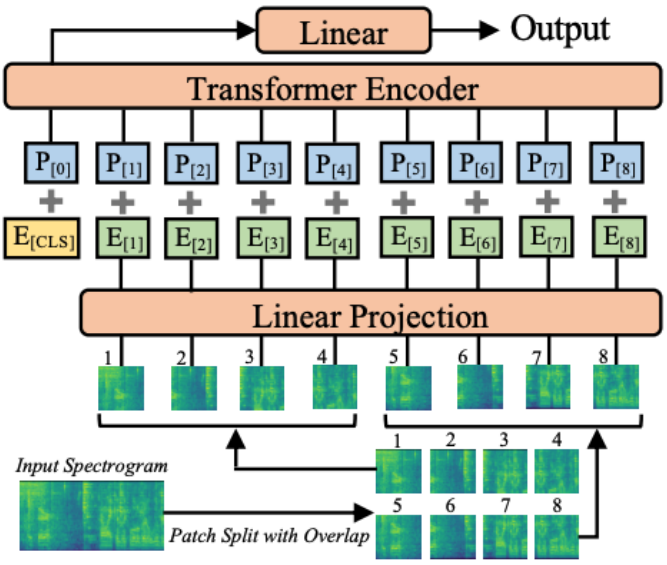

Figure 1.16: AST. Picture from the original paper [2].

Using AST's way of extracting spectrogram patches is integral to SpectroBERT. It allows us to pick up on subtle aspects of speech, like tone and rhythm, which are fundamental for getting the full picture of spoken language. These patches are then turned into the tokens that SpectroBERT works with, with the goal that all the rich details in the audio are used to the fullest and matched up effectively with text data.

1.3.3 Wav2Vec2

Wav2Vec2 [16] [17], developed by Facebook AI, is a state-of-the-art model for learning representations from raw audio data. Its architecture is designed to capture hidden aspects of speech without needing labels, thanks to its unsupervised pre-training, making it excel in a variety of audio-related tasks. Wav2Vec2's strength lies in its encoder, which transforms raw audio waveforms into a latent space that efficiently captures the intricate details of sounds in a compact and meaningful representation.

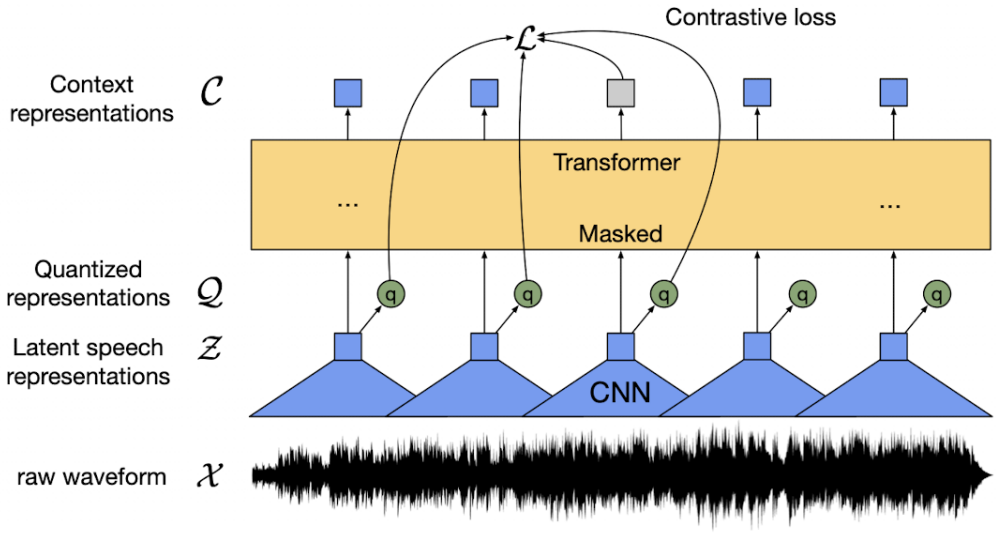

Figure 1.17: Wav2Vec2. Picture from the original paper [16].

In the development of SpectroBERT, we decided to incorporate several layers from Wav2Vec2 to process the rich spectrogram patches previously obtained trough AST's preprocessing, into a format that is compatible with the BERT [3] text encoder. By utilizing layers from Wav2Vec2, we turn the each spectrogram into an 'audio token' - a feature vector with the same dimensionality as the textual features. By leveraging Wav2Vec2's layers for audio encoding, SpectroBERT benefits from

the robust feature extraction that Wav2Vec2 offers, setting a solid foundation for the subsequent cross-modal learning tasks.

1.3.4 Audio-Language Modeling

This section introduces recent key developments in audio-language modeling, focusing on CM-BERT [18], SpeechBERT [19], and CTAL [20]. Each of these models takes a different approach to combining audio and text, and we will briefly outline how they work and how their proposed approach differs from ours. Understanding these works also clarifies how SpectroBERT fits into this wider landscape of audio-text integration.

CMBERT

CM-BERT [18], also known as Cross-Modal BERT, is an innovative approach in the audio-text integration domain, aiming to effectively merge spoken language with its textual counterpart. It operates on a dual-encoder framework where separate encoders are dedicated to audio and text inputs. Each encoder is responsible for creating embeddings, which are essentially rich, contextual representations of the respective modalities. These embeddings are then aligned, enabling CM-BERT to develop a cohesive understanding of the content from both audio and textual perspectives.

In contrast, our model, SpectroBERT, takes a different route by adapting the VisualBERT [1] framework and substituting visual tokens with audio spectrogram patches. This adaptation allows for a more seamless integration of the audio and text modalities within a singular, transformer-based architecture, diverging from CM-BERT's approach of using two distinct encoders. SpectroBERT's methodology is designed to not only process but also intertwine audio and text data, leveraging the self-attention mechanism to understand and align the cross-modal information within a unified context. This architectural distinction gives SpectroBERT an edge, in terms of simplicity, in handling speech-text cross-modal learning, offering a more integrated perspective compared to the separate processing paths employed by CM-BERT.

SpeechBERT

SpeechBERT [19] uniquely combines audio and text by initially aligning audio word embeddings with text word embeddings from BERT [3], creating a unified phonetic-semantic representation. Like BERT, it undergoes MLM pre-training with both audio and text, which enhances its ability to process spoken language,

especially for Spoken Question Answering (SQA). Its design is particularly effective in handling tasks where Automatic Speech Recognition (ASR) errors are common, as it directly works with audio inputs for answer span identification.

SpectroBERT, in contrast, directly integrates audio data as spectrogram patches into a transformer-based architecture, following the approach of VisualBERT. This method allows us to process audio and text in a more integrated manner, leveraging self-attention for deeper audio-text interaction. Unlike SpeechBERT, which is focused on SQA, SpectroBERT's versatility allows it to adapt to various tasks like Audio Question Answering and Speech Emotion Recognition by modifying the model's heads.

In summary, while SpeechBERT excels in SQA by enhancing spoken language understanding, SpectroBERT provides a more flexible approach to audio-text integration, suitable for a wider range of applications.

CTAL

CTAL [20], or Cross-Transformers for Audio-and-Language, marks another significant effort in the fusion of audio and text modalities. This model uniquely leverages a transformer-based approach to process audio and text data in parallel, employing separate but interconnected transformer networks for each modality.

In CTAL, the audio is first transformed into a feature representation, which is then processed alongside the text data through these cross-modal transformers. This setup allows CTAL to capture the distinct characteristics of both speech and text while facilitating interaction between the two, making it effective for tasks that require a deep understanding of the combined audio-text information.

SpectroBERT, in contrast to CTAL, adopts a different architectural strategy. Instead of parallel transformers for each modality, SpectroBERT integrates audio information directly into the transformer model used for text, by converting audio into spectrogram patches. This method ensures that the audio and text data are not just processed in parallel but are intricately mixed together within the same transformer layers. Our approach emphasizes a more unified processing of the two modalities, using a single transformer model to align and understand the speech and text data in a shared context, offering a more cohesive and streamlined process for handling cross-modal data, differentiating it from CTAL's dual-transformer structure.

Chapter 2

Combining Speech and Text Language Models

2.1 Introduction to SpectroBERT

In our efforts to create a model that seamlessly integrates sound and text, we developed SpectroBERT. Our model aims to be a step forward from the VisualBERT framework, reimagined to handle audio data effectively.

SpectroBERT starts with the basic structure of VisualBERT [1] but takes a different turn by focusing on audio rather than visual data. We've replaced VisualBERT's image processing component with the audio processing inspired by AST [2]. This change lets SpectroBERT handle audio in a similar way to how VisualBERT deals with images.

SpectroBERT converts audio into 'audio tokens', which are then paired with text data, forming a combined sequence that passes through the transformer [4]. Here, the model's self-attention mechanism shines, finding and aligning interactions between the sound and text elements, resulting in a model that understands both speech and its written form.

Like its predecessor, we don't want SpectroBERT to be a static entity but rather a versatile framework, adaptable to a multitude of downstream tasks, like audio question answering or sentiment analysis, by simply changing the model's 'head' - the components that come after the transformer layers.

In the following sections, we'll dive deeper into how SpectroBERT works, from processing raw audio to its application in different speech-text language tasks, starting from its architecture.

2.2 SpectroBERT Architecture

The architecture begins with the preprocessing of audio inputs, employing AST's methodology to extract spectrogram patches, that serve as the visual representation of sound, capturing the nuances of audio frequencies and temporal patterns in a form that can be processed by neural networks. From here, the journey of an audio sample in SpectroBERT transitions to encoding, where layers taken from Wav2Vec2 [16] encode the spectrogram patches into discrete tokens, similar to the process by which text is tokenized before being fed into BERT [3].

These audio tokens are then concatenated with their textual counterparts, creating a sequence that is then processed by SpectroBERT's transformer layers. The self-attention mechanism within these layers is the core of the architecture, as this it what allows the model to draw connections and learn from the interaction between the audio and text tokens.

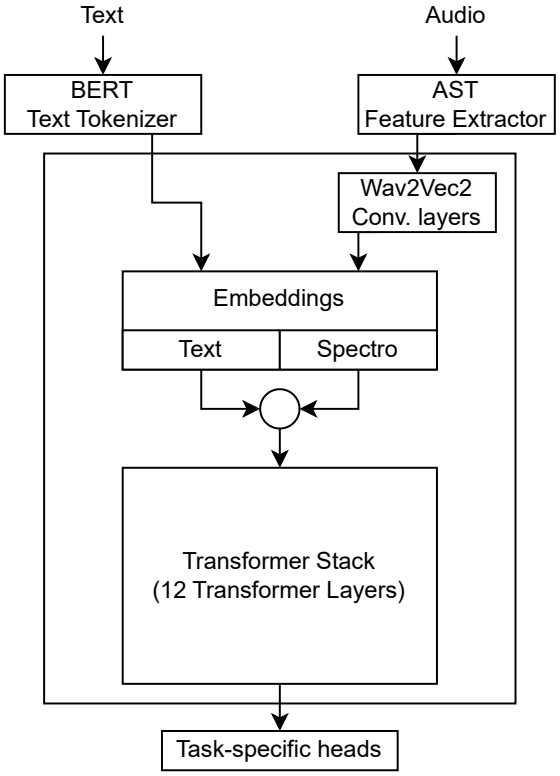

Figure 2.1: SpectroBERT Architecture

2.2.1 Audio Preprocessing and Spectrogram Patch Extraction

SpectroBERT begins by first processing the audio, which involves turning complex sounds into a structured format that's suitable for deep learning. This transformation draws from techniques developed by AST, which cleverly converts time-domain audio signals into spectrogram patches - rich visual representations that encapsulate the frequency, time, and amplitude information of the audio signal, much like a visual snapshot of sound.

The preprocessing pipeline begins with breaking down the audio through methods like the Short-Time Fourier Transform [21] [22] (STFT) or similar techniques, which split the continuous audio signal into short segments and map them into the frequency domain. The outcome is a two-dimensional spectrogram that's laid out like a graph, where the x-axis represents time, the y-axis represents frequency, and the intensity of each point reflects the amplitude or energy at that particular time and frequency.

Next, the spectrogram is sliced into overlapping patches, a strategy that ensures no loss of contextual information at the edges of each piece. These patches are similar to the image regions VisualBERT works with, and they are critical for capturing the local and global dependencies within the audio data. Each patch is treated as a separate entity, representing a specific temporal slice of the audio, and is subsequently encoded into an 'audio token' ready for the model to analyze.

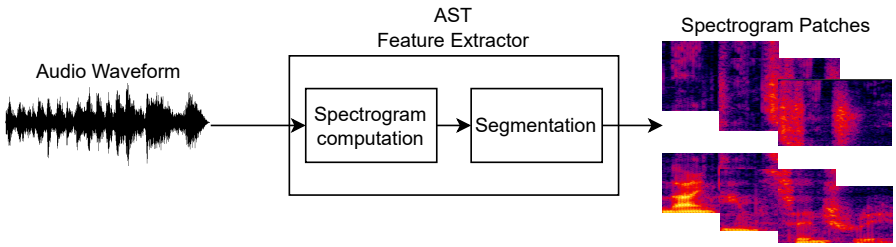

Figure 2.2: Audio Preprocessing and Spectrogram Patch Extraction

2.2.2 Audio Encoding with Wav2Vec2 layers

In SpectroBERT, we turn spectrogram patches into a series of tokens ready for analysis, a process made possible by using layers borrowed from Wav2Vec2 [16], which are key to transforming the detailed patterns in the spectrograms into a form that the Transformer [4] architecture can understand and work with.

Each spectrogram patch, rich with acoustic detail, goes through a series of convolutional layers that act as feature extractors, applying a series of convolutions that gradually refine the raw audio data into a set of feature representations. The convolutional nature of these layers allows them to capture local relations within the patches, identifying characteristic audio signatures that are essential for understanding the speech in the recordings.

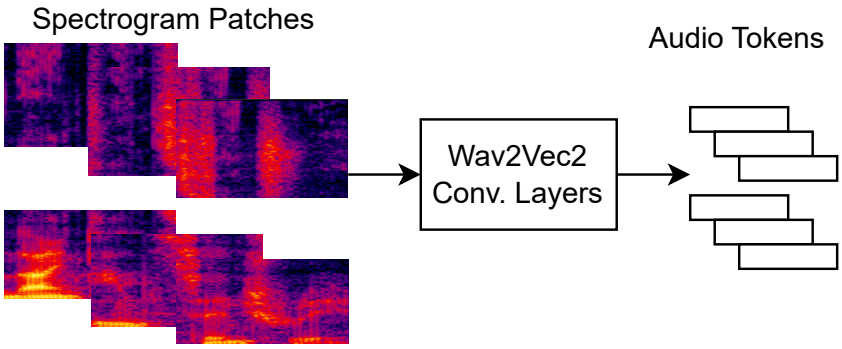

Figure 2.3: Audio Encoding. By means of Wav2Vec2 convolutional layers, each patch is reduced into a 'audio token'.

The extracted features from these convolutional layers are then passed through a series of non-linear transformations, ultimately resulting in a set of audio tokens, which encapsulate the essence of the spectrogram patches into a dense, neural network-friendly format. This encoding process ensures that the tokens carry meaningful patterns of the speech, which are fundamental for the subsequent stages of processing within SpectroBERT.

We've made sure that these audio tokens match up dimensionally with the text tokens processed by the BERT text encoder. This dimensional parity allows the audio and text data to be combined smoothly in the Transformer layers, ensuring that the self-attention mechanism works well across both types of data.

2.2.3 Text Preprocessing and Encoding into Tokens

Text preprocessing begins with tokenization, where the input text is segmented into atomic elements that could be as small as words, subwords, or even characters. This is achieved using a tokenizer that's trained to pick up on the linguistic patterns in the text, often relying on data compression techniques like Byte Pair Encoding [23] (BPE) - as done in the BERT's tokenizer we used. The goal of this step is to have a tokenizer that's smart enough to handle common and rare words without needing an excessively large vocabulary, but rather building complex and long words by

combination of simpler and shorter components.

Following tokenization, each token is mapped to a unique integer ID, creating a sequence of numerical representations that correspond to the input text. These IDs are then used to retrieve embeddings from a pre-trained lookup table, which contains dense vector representations that carry information about each token, like what it means and how it's used in language.

The final step in text preprocessing involves the addition of special tokens to the sequence, that denote the beginning, end, and separation of sentences. For instance, BERT[3] introduces special tokens such as '[CLS]' for classification tasks and '[SEP]' to separate multiple sentences. These tokens serve specific functions during model training and inference, helping the Transformer's self-attention mechanism to understand the input structure correctly.

The mathematical representation of this encoding process into token embeddings, including the special tokens, can be expressed as follows:

$$\mathbf{E}_{\text{token}} = \text{EmbeddingLookup}(ID_{\text{token}})$$

Here, $\mathbf{E}_{\text{token}}$ is the embedding of a token, and ID_{token} is the token's unique identifier obtained post-tokenization.

2.2.4 Concatenation of Audio and Text Modalities

The audio tokens, which we get as output from the convolutional layers borrowed from Wav2Vec2, carry detailed information about the sounds. On the other hand, the text tokens, processed by BERT's text encoder, contain rich linguistic information.

In combining them before being fed to the sequence of Transformer layers, we simply concatenate the tokens from the two modalities, making sure that additional information like padding masks and sequence identifiers are included.

2.2.5 The Transformer Stack and Self-Attention Mechanism

At the heart of SpectroBERT lies the Transformer [4] stack, a sequence of Transformer layers that gives the model its capacity for deep contextual understanding. The stack is composed of multiple layers of self-attention mechanisms and position-wise feed-forward networks, which work together to process the concatenated sequence of audio and text tokens, enabling the capture of subtle interactions within and across the different modalities.

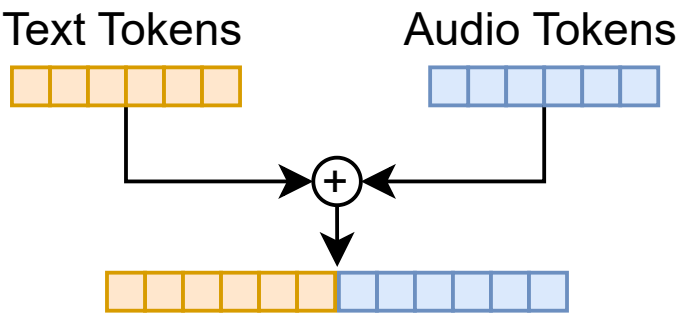

Figure 2.4: Concatenation of Audio and Text Modalities

The self-attention mechanism is the key feature of the Transformer stack, allowing SpectroBERT to weigh the importance of each token in relation to others in the sequence. It computes attention scores that determine how much focus should be given to other tokens when processing a particular one, or in human words *"When you're looking at this word or sound, you should also consider these other words or sounds to this extent"*.

Mathematically, the self-attention for a single head can be formulated as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

In this formula, Q, K, and V stand for the query, key, and value matrices derived from the input tokens, d_k represents the dimension of the keys, and the softmax function is applied to the scaled dot-product of Q and K, determining how much weight to put on the different values V.

Through self-attention, SpectroBERT can perform a nuanced analysis that takes into account the full context of the sequence. Each layer in the Transformer stack refines this analysis, allowing for increasingly sophisticated representations to emerge as the data moves through the model.

This stack is not just a one-size-fits-all solution; but rather it's tuned during both the pre-training and fine-tuning stages, adapting to the patterns and specific kind of the multimodal dataset it sees. In short, the Transformer stack is where the real strength of SpectroBERT lies, because it's here that audio and text come together.

2.2.6 Model Heads for Downstream Tasks

In the final layer of SpectroBERT's architecture lies the model heads, which are neural network components tailored for specific downstream tasks.

Each task requires its own approach to how the model's outputs are interpreted and utilized, and must be designed to interface seamlessly with the Transformer stack, receiving the contextually-enriched token sequences and applying the final layers to produce task-specific results.

For instance, a classification head may consist of a softmax layer that maps the Transformer stack's output to a set of class probabilities, enabling SpectroBERT to categorize inputs. This can be represented as:

$$P(\text{class}|\text{input}) = \text{softmax}(W_{\text{head}} h_{\text{last}} + b_{\text{head}})$$

Where $P(\text{class}|\text{input})$ represents the conditional probability of a class given the input, W_{head} and b_{head} are the weights and bias of the head layer, and h_{last} is the last layer's output from the Transformer stack.

The design of model heads in SpectroBERT is modular, allowing to attach and train heads that best suit the end-goals, without the need to retrain the entire network.

2.3 Pre-training SpectroBERT

2.3.1 Introduction to Pre-training

The pre-training stage of SpectroBERT is an adaptation of the successful strategies employed by VisualBERT [1], but tailored to the auditory domain. By shifting the focus from visual to auditory, SpectroBERT is taught to understand the relationship between spoken language and corresponding text.

A key part of this pre-training is Masked Language Modeling (MLM), a method originally used in BERT [3]. In this process, we randomly hide some of the text tokens in a sentence and ask the model to guess these tokens based on the unmasked surrounding text and the full audio input. Unlike the visual tokens in VisualBERT that match with specific image regions, the audio tokens in SpectroBERT represent segments of sound, which remain unmasked and serve as additional context that can help in predicting the masked textual tokens.

Alongside MLM, we also employ a Sentence-Audio Prediction (SAP) objective, similar to BERT's Next Sentence Prediction (NSP). In this task, the model looks at pairs of sentences and audio segments to predict whether the audio matches the

text. With a balanced probability - 50% that the audio matches the text and 50% that it's a random sample from the dataset - the model learns how elements of the spoken and written language *'appear'* together.

The pre-training process for SpectroBERT, which leverages these dual objectives, is key in obtaining a robust model capable of understanding and integrating information across speech and text. Unlike some models where all types of data might be masked during training, we keep the audio visible to aid the learning process, thus facilitating a deeper multimodal comprehension.

In the following sections, we will delve into the specifics of the MLM and SAP objectives, including the kind of loss functions used and how the model's architecture handles these tasks during pre-training. Additionally, we will introduce the datasets we propose for pre-training stage, although more details will follow in the next Chapter 3. This pre-training stage is essential for SpectroBERT to learn a prior understanding of the nuances of language and sound, setting a solid foundation for the following fine-tuning stage, where the model is specialized for specific downstream tasks.

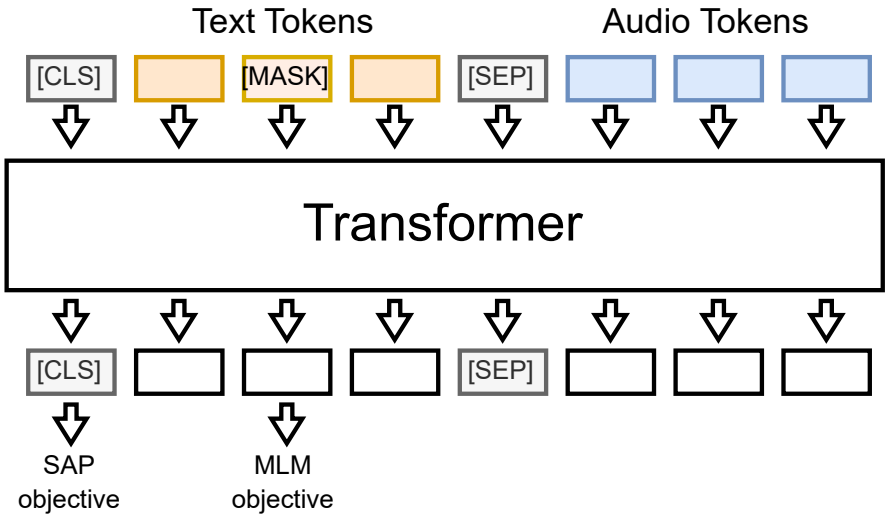

Figure 2.5: Pre-training

2.3.2 Masked Language Modeling (MLM)

The Masked Language Modeling (MLM) is a crucial part of SpectroBERT's pre-training, where the model learns about language structure and meaning.

In this objective, a portion of the input text tokens is randomly selected and

replaced with a [MASK] token, and the goal for SpectroBERT is to predict the original token that has been masked out, relying on the context provided by the surrounding unmasked textual tokens and the accompanying audio data.

This task of guessing the hidden words helps SpectroBERT get better at understanding language from all directions, muck like BERT [**bert**], because it has to consider the words before and after the masked token and use clues from the audio, which serves as a complementary source of information. When using pairs of text and its spoken version, the inclusion of audio data is particularly interesting, as it allows SpectroBERT to utilize additional cues from the spoken version to potentially capture nuances like emphasis and tone that are not present in the text alone.

We follow the typical implementation of MLM, so approximately 15% of the tokens in each sequence are masked. The selection of these tokens is random but follows a uniform distribution to ensure that all tokens have an equal probability of being masked throughout the pre-training data. The MLM objective can be formally defined using the Cross Entropy Loss function as:

$$L_{\text{MLM}}(\theta) = -\sum_{i \in \text{masked}} \log p_\theta(\text{token}_i | \text{context}_i)$$

In this formula, $L_{\text{MLM}}(\theta)$ represents the total loss for the masked tokens, $p_\theta(\text{token}_i|\text{context}_i)$ is the probability assigned by the model with parameters θ to the correct token token_i given the context context_i, which includes both the surrounding text and audio tokens, and the sum is taken over all masked tokens.

During pre-training, the model parameters are adjusted to minimize this loss across all masked tokens, which improves SpectroBERT's ability to predict the masked words accurately.

2.3.3 Sentence-Audio Prediction (SAP)

Alongside learning about words and their meanings, SpectroBERT also gets trained to understand how sentences and audio clips are related. This is done through the Sentence-Audio Prediction task, which is adapted from the Next Sentence Prediction (NSP) task used in BERT, but with a twist to include audio.

In this task, SpectroBERT is presented with pairs of text and audio data, and the model has to figure out if the given audio segment correctly corresponds to the text. During pre-training, half of the input examples are paired with their matching audio segments, creating a positive example, while the other half is paired with audio segments taken randomly from the other samples in the dataset, creating a negative example. This reduces to a binary classification task, that essentially

forces the model to learn contextual embeddings that are rich and informative enough to discern the correct pairings of text and audio.

The Sentence-Audio Prediction task can be expressed through a Binary Cross Entropy Loss, which for a single example is given by:

$$L_{\text{SAP}}(\theta) = -y \log(p_\theta) - (1 - y) \log(1 - p_\theta)$$

Here, $L(\theta)$ represents the loss we're trying to minimize, y is the true label (1 if the audio matches the text, 0 if not), and p_θ is the probability predicted by the model with parameters θ that the audio matches the text. The goal during pre-training is to get this loss as low as possible, so the model learns to predict when text and audio are talking about the same thing.

2.3.4 Cross Entropy Loss in Pre-training

The Cross Entropy Loss function is a critical component in the pre-training of SpectroBERT, serving as the primary metric that tells us how well the model is doing in the Masked Language Modeling (MLM) and Sentence-Audio Prediction tasks. Basically, it measures the gap between what the model predicts and the actual answers (or labels) we have, by quantifying the difference between the predicted probabilities and the actual distribution of the labels.

For the MLM task, we use Cross Entropy Loss to check each masked token, measuring the model's ability to predict the correct token from the entire vocabulary. The loss for a single masked token is calculated as the negative log likelihood of the correct token's probability:

$$L_{\text{MLM}}(\theta) = -\sum_{i \in \text{masked}} \log p_\theta(\text{token}_i | \text{context}_i)$$

Similarly, for the Sentence-Audio Prediction task, the Cross Entropy Loss evaluates the model's ability to tell whether the audio matches the text. Since this is a yes-or-no question, the loss calculation is a bit simpler:

$$L_{\text{SAP}}(\theta) = -y \log(p_\theta) - (1 - y) \log(1 - p_\theta)$$

The overall loss for pre-training SpectroBERT is a weighted sum of these two losses:

$$L_{\text{total}} = \alpha L_{\text{MLM}}(\theta) + \beta L_{\text{SAP}}(\theta)$$

The weights α and β balance the contribution of each task to the total loss, allowing the model to prioritize learning from the MLM and Sentence-Audio Prediction tasks according to the specific requirements of the downstream applications.

In our setup, they are both set to 0.5, meaning equal importance.

Minimizing the L_{total} across the entire pre-training dataset ensures that SpectroBERT learns a generalized representation of language and audio-text alignment.

2.3.5 Pre-training Data and Strategies

For SpectroBERT to learn effectively during pre-training, it's important to select rich and diverse datasets. We suggest the usage of three main ones: LibriSpeech [24], Common Voice [25], and WavCaps [26], that have been chosen because each of these datasets brings something different to the table, given their different nature.

- LibriSpeech is packed with English speech from audiobooks, which means it offers a wide variety of speaking styles and stories, that is great for exposing the model to different ways people can express themselves.
- Common Voice, on the other hand, is more about everyday speech. It's a crowd-sourced repository that offers lots of accents and dialects, reflecting more conversational and less formal speech, like the language you would hear in regular every-day conversations.
- WavCaps instead shakes things up by bringing a more diverse set of audio, that is not limited to speech, and with its different sound qualities and background noises, challenges the model to adapt to various audio elements.

These datasets can serve as the groundwork for SpectroBERT's pre-training, by exposing the model to a broad range of different linguistic and acoustic scenarios. Moreover, we suggest that by pre-training on these different sources, SpectroBERT could learn to pick up the subtle differences between datasets in different ways, potentially leading to varying performances on downstream tasks. This highlights the importance of dataset selection in pre-training, as it can significantly influence the model's generalization and adaptability.

More details on these datasets will follow in Chapter 3.

2.4 Fine-tuning SpectroBERT

2.4.1 Introduction to Fine-tuning

After the extensive pre-training phase, it's time to move on to the fine-tuning stage, where SpectroBERT is appropriately adapted to perform specific tasks, using the

broad knowledge it gained during pre-training and refining it for particular uses, by adjusting the model weights to make accurate predictions for downstream tasks. In our case, we're focusing on two main tasks: Audio Question Answering and Emotion Recognition.

During this phase, we make changes mainly to the final parts of the model, which we previously referred to as the 'heads'. This way, SpectroBERT can shift its attention from the wide-ranging understanding it developed earlier to more specific details that are key for doing well in the chosen tasks.

The fine-tuning process involves smaller, more focused datasets, often annotated to provide ground truth, which allows for supervised learning, where the model can really focus and get better at the particular challenges of each task. The idea is to use the solid foundation acquired during pre-training and build upon it, refining the model's abilities so that it can not only understand language and audio but also apply this understanding in practical, real-world scenarios.

2.4.2 Generic Classification Task

Fine-tuning SpectroBERT for classification involves appending a classification layer - or head - onto the pre-trained model. This head typically consists of a linear layer followed by a softmax function, transforming the representation produced by the Transformer [4] stack into a probability distribution over the target classes:

$$P(y|\mathbf{x}) = \text{softmax}(W_c \cdot h_{\text{CLS}} + b_c)$$

Here, $P(y|\mathbf{x})$ is the predicted probability of class y given the input $\mathbf{x}$, W_c is the weight matrix, b_c is the bias vector for the classification layer, and h_{CLS} is the final hidden state corresponding to the special [CLS] token from SpectroBERT's output sequence.

During fine-tuning, SpectroBERT'weights are adjusted to minimize the Cross Entropy Loss between the predicted and actual class labels:

$$L_{\text{CLS}} = -\sum_{i=1}^{N} \log P(y_i|\mathbf{x}_i)$$

In this loss function, N is the number of training examples, y_i is the true class label, and $\mathbf{x}_i$ is the input for the i-th example.

2.4.3 Audio Question Answering

Audio Question Answering [27] (AQA) is a cutting-edge area in multimodal learning, and it's a real test for models like SpectroBERT to provide accurate natural language responses based on both audio inputs and textual questions. This task really pushes the model towards a deep cross-modal comprehension, as it must learn to discern relevant information from the audio and align it with the question posed in text form.

To adapt SpectroBERT for AQA, we switch out the heads used during pretraining, that are replaced with a classification head. This head is as a single linear layer tailored to the AQA task's requirements, whether it's figuring out a binary yes/no answer or selecting from multiple choice options in open-ended questions. For tasks where the answer is either 'yes' or 'no', we use just one output neuron, and apply a sigmoid function to this neuron's output to interpret it as the likelihood of a 'yes' answer:

$$P(\text{yes}|\mathbf{x}, \mathbf{q}) = \sigma(W_{\text{AQA}} \cdot h_{\text{CLS}} + b_{\text{AQA}})$$

In this equation, $P(\text{yes}|\mathbf{x}, \mathbf{q})$ is the probability of a 'yes' answer given the audio input $\mathbf{x}$ and the text question $\mathbf{q}$. σ denotes the sigmoid function, W_{AQA} is the weight matrix, b_{AQA} is the bias term for the AQA head, and h_{CLS} is the final hidden state corresponding to the special [CLS] token from the Transformer stack's output.

For the 'open' variant, the setup is similar, but instead of a single neuron, we use multiple output neurons - one for each possible answer word. The activation of these neurons is then passed through a softmax function as part of the Cross Entropy Loss during training, which allows the model to pick the most likely answer from the available options:

$$P(\text{word}_i|\mathbf{x}, \mathbf{q}) = \text{softmax}(W_{\text{AQA}} \cdot h_{\text{CLS}} + b_{\text{AQA}})_i$$

Performance on the AQA task is a strong indicator of the quality of the learned audio-text representations. We use the Clotho-AQA [27] dataset for training, because it provides both binary and single-word answer questions.

2.4.4 Speech Emotion Recognition

Speech Emotion Recognition [28] (SER) is an intricate challenge where SpectroBERT is fine-tuned to interpret and categorize the emotional tones in spoken language. SER goes beyond a mere analysis of *what* was said, the content, but it's rather about picking up on *how* something was said, in terms of speech patterns —

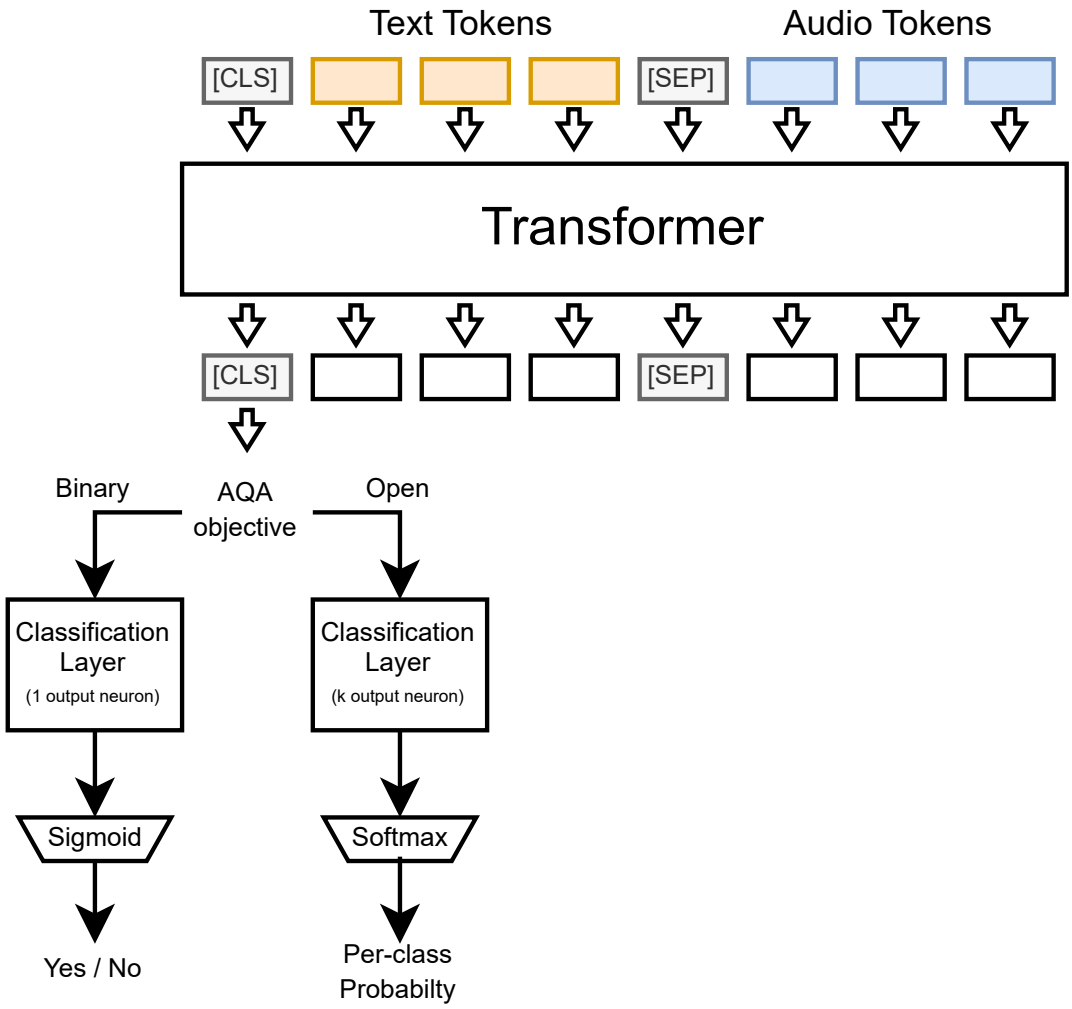

Figure 2.6: Audio Question Answering Tasks

tone, pitch, and rhythm — which are indicative of a speaker's emotions such as happiness, anger, sadness, or frustration.

In fields like Human-Computer Interaction (HCI) and advanced speech processing systems, SER is an increasingly important task. The complexity of detecting the presence of various emotions in the speaker's voice stems from the nuanced and subjective nature of emotional expression - we are basically asking machines to quantify what is inherently a qualitative human experience.

For SpectroBERT, adapting to the SER task is similar to how we handled the AQA task. The pre-training heads are replaced with a classification head, this time

focusing on identifying emotions. This head consists of a linear layer with a number of output neurons corresponding to the number of emotions we want to detect. Just like with AQA, we use a softmax function to turn the neuron activations into a probability distribution for each emotion:

$$P(\text{emotion}_i|\mathbf{x}) = \text{softmax}(W_{\text{SER}} \cdot h_{\text{CLS}} + b_{\text{SER}})_i$$

In this equation, $P(\text{emotion}_i|\mathbf{x})$ represents the probability of emotion i given the speech input $\mathbf{x}$. W_{SER} and b_{SER} are the weights and biases term for the SER head, and h_{CLS} is the final hidden state corresponding to the [CLS] token from SpectroBERT's output, capturing the overall emotional tone of the input.

To fine-tune SpectroBERT for SER, we need a dataset with audio clips labeled with their emotional states, and for that, we suggest RAVDESS [29] and IEMOCAP [30], which will also be discussed in Chapter 3.

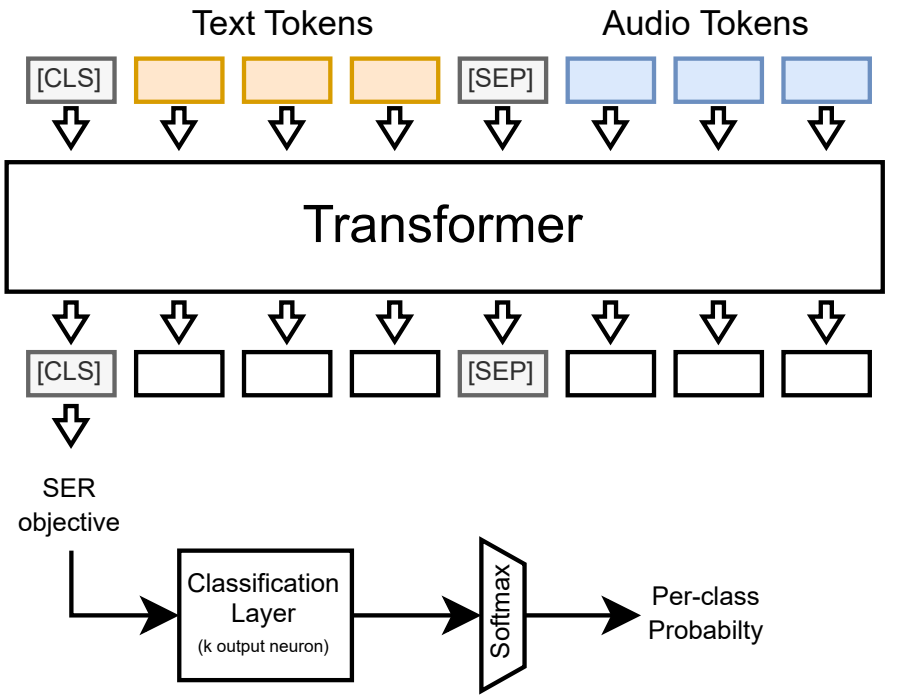

Figure 2.7: Speech Emotion Recognition

2.4.5 Other Possibilities for Fine-Tuning

SpectroBERT's flexible design means it is not only limited to classification tasks like Audio Question Answering and Speech Emotion Recognition. Its ability to

process and combine audio and text data also makes it a good candidate for sequence-to-sequence [31] (Seq2Seq) tasks too, that are tasks about generating output sequences based on input sequences, which can be quite complex.

In the case of SpectroBERT, Seq2Seq tasks could look like speech translation, where the model turns spoken words in one language into written text in another. Or it could be about summarizing what's spoken, boiling down lengthy audio to just the key points in a written format. Another possibility is using SpectroBERT in dialogue systems to come up with text responses to spoken queries, almost like having a conversation.

To tackle these kinds of tasks, SpectroBERT would be fine-tuned with an extra layer designed for generating sequences. Imagine a decoder that puts out one piece of the sequence at a time, building up a response or translation bit by bit, understanding and maintaining the flow of a conversation or narrative, which can be pretty challenging.

Chapter 3

Datasets

This chapter is dedicated to the datasets that we suggest using for a study of SpectroBERT. Since the quality of a machine learning model is deeply influenced by the data it is trained on, a careful selection of pre-training and fine-tuning datasets is essential. We've chosen these datasets not just for their rich mix of audio and text but also because they present the model with a wide range of real-world language use, which is crucial for testing and improving its abilities.

In the following, we detail the datasets suggested for the pre-training and fine-tuning stages, their characteristics, and the reasons for their selection.

3.1 Pre-training

3.1.1 Librispeech

LibriSpeech [24] is a corpus of read English speech designed to provide resources for the training and evaluation of Automatic Speech Recognition (ASR) systems. This dataset is derived from audiobooks in the LibriVox project, amounting to roughly 1000 hours of spoken English. Recorded at a sampling rate of 16 kHz, LibriSpeech offers high-quality audio data that has been segmented and meticulously aligned with transcripts.

The dataset is structured into subsets, catering to varying levels of modeling complexity: 'train-clean-100', 'train-clean-360', 'train-other-500' for training, and 'dev-clean', 'dev-other', 'test-clean', and 'test-other' for development and testing. The distinction between 'clean' and 'other' refers to the recording quality and speaker's accent, with 'clean' representing higher fidelity and less accented speech.

In addition to audio recordings, LibriSpeech provides extensive language resources, including n-gram models and the text of read passages from Project Gutenberg books. This textual component is substantial, comprising over 803 million tokens and nearly 977 thousand unique words, enriching the dataset's utility for language modeling tasks.

LibriSpeech's versatility has been demonstrated through its widespread use across various speech-related research endeavors. It has played a crucial role in pushing forward the fields of Speech Recognition, Speech Enhancement, Voice Conversion, and more. The expansive nature and comprehensive structure of LibriSpeech make it an invaluable asset for pre-training SpectroBERT, offering a broad linguistic landscape from which the model can learn a deep representation of spoken English language, crucial for its later abilities in cross-modal understanding.

3.1.2 Common Voice

Common Voice [25], curated by Mozilla, stands out as a monumental voice dataset distinguished by its open-source nature and multilingual diversity. As a crowd-sourced compilation, it harnesses the collective contributions of volunteers worldwide, offering an ever-expanding repository of recorded speech.

This dataset is a reflection of real-world linguistic diversity, encompassing a broad spectrum of accents, ages, and gender demographics. Each audio recording is paired with its corresponding textual transcription, providing an aligned dataset that is perfect for development of speech recognition technology, and the inclusion of demographic metadata adds another layer of richness, enabling the training of more fair speech recognition models.

Its dynamic nature means that it is in a constant state of growth, with new voice recordings and languages continuously augmenting the dataset. Researchers and developers can access Common Voice in its entirety or in segmented versions, which include "Delta Segments" for efficient integration of the latest recordings, ensuring that users can stay up-to-date with the freshest data without the need to re-download the entire corpus.

The availability and scale of Common Voice make it an unparalleled resource for building voice-enabled technologies. It democratizes access to a vast range of voice data, fostering innovation and competition in the development of machine learning applications centered around speech.

3.1.3 WavCaps

WavCaps [26] emerges as a pivotal dataset tailored for audio-language multimodal research, especially in the realm of audio captioning. WavCaps introduces approximately 400,000 audio clips, each accompanied by a paired caption, forming a substantial foundation for training models that can understand and describe audio content.

The dataset's audio clips are curated from diverse web sources — FreeSound, BBC Sound Effects, SoundBible — and supplemented with the AudioSet strongly-labelled subset for a comprehensive range of sound events, ensuring a rich variety of acoustic environments and scenarios, crucial for developing a model capable of generalizing across different auditory contexts.

Given the inherently noisy nature of web-sourced data, WavCaps employs a sophisticated three-stage processing pipeline to refine and elevate the quality of its captions. This pipeline uses advanced language models like ChatGPT to filter out noise and enhance the descriptions automatically, making them more coherent and contextually relevant for training purposes.

Training systems on WavCaps have shown marked improvements over previous state-of-the-art models, proving the dataset's quality and the effectiveness of its processing pipeline. It has proven to be a valuable asset in advancing audio-language multimodal learning, providing a solid benchmark for models to strive towards.

3.2 Fine-Tuning

3.2.1 Clotho-AQA

Clotho-AQA [27], a dataset explicitly crafted for Audio Question Answering (AQA) offers a focused environment to test and refine AQA systems, a subset of tasks requiring nuanced comprehension of audio content and its relevance to posed questions.

Comprising 1,991 audio clips sourced from the larger Clotho dataset, ClothoAQA presents a challenging array of soundscapes, each accompanied by a set of six questions with corresponding answers. The audio clips, ranging from 15 to 30 seconds, offer a snapshot of diverse acoustic environments, providing a rich testing ground for SpectroBERT's interpretative capabilities.

The dataset's questions are categorized to specific types of responses: two are

binary (yes/no answers), and four are single-word answers, all of which are obtained via crowdsourcing platforms like Amazon's Mechanical Turk. This methodology ensures a wide array of question types and answer styles, reflecting the variability inherent in natural language inquiries and responses.

The ClothoAQA dataset not only allows us an evaluation of SpectroBERT's performance on binary and single-word answer tasks but also serves as a benchmark for future innovations in AQA. The baseline experiments provided by the authors, utilizing LSTM-based classifiers, set preliminary performance standards with an accuracy of 62.7% for the binary classifier and a top-1 accuracy of 54.2% for the multi-class classifier, with a top-5 accuracy reaching 93.7%.

3.2.2 RAVDESS

The Ryerson Audio-Visual Database of Emotional Speech and Song [29] (RAVDESS) is a meticulously curated multimodal dataset that serves as a cornerstone for research in emotional recognition. It is a comprehensive collection of high-quality audio and video recordings focused on capturing a wide range of emotions through vocal and facial expressions, articulated in North American English.

With a total of 7,356 files, RAVDESS showcases the talents of 24 professional actors — equally divided between female and male performers — who vocalize lexically-matched statements designed to be emotionally evocative. The inclusion of professional actors ensures a degree of consistency and clarity in the emotional expressions, enhancing the dataset's reliability for training purposes.

The dataset encompasses a spectrum of emotions, including calm, happiness, sadness, anger, fear, surprise, and disgust, articulated with varying degrees of intensity. This variation allows for the exploration of not only the type of emotion being expressed but also its intensity, providing a nuanced understanding of emotional expression in speech.

RAVDESS's versatility makes it an ideal dataset for fine-tuning SpectroBERT on Speech Emotion Recognition (SER) tasks. The clear categorization of emotions enables SpectroBERT to develop a sophisticated model of emotional expression, learning to discern subtle differences in tone, pitch, and rhythm associated with each emotional state.

The dataset's application extends beyond SER; it has been employed in tasks ranging from facial emotion recognition to emotion classification, both in audio and visual modalities.

3.2.3 IEMOCAP

The Interactive Emotional Dyadic Motion Capture [30] (IEMOCAP) dataset, assembled by the Signal Analysis and Interpretation Laboratory (SAIL) at the University of Southern California (USC), represents a rich multimodal collection of emotional expression data. Collected in a controlled environment with a focus on capturing genuine affective interactions, IEMOCAP is an acted dataset that offers researchers high-fidelity insights into the dynamics of human emotion.

Spanning approximately 12 hours, the dataset incorporates multiple data modalities, including video, speech, facial motion capture, and text transcriptions. It features 151 dialogues, each a dyadic interaction between two actors, culminating in 302 unique dialogue videos. The dyadic format is a deliberate choice, designed to elicit a range of emotions through naturalistic exchanges between the participants.

The term "dyadic" is derived from the word "dyad", that means a pair, a group of two, referring to the fact that dialogues in the dataset happen in pairs. The actors in IEMOCAP were tasked with performing both improvised and scripted scenarios, carefully chosen to provoke a wide spectrum of emotional states. The resulting performances were annotated for nine distinct emotions: anger, fear, excitement, sadness, happiness, surprise, frustration, disappointment, and neutrality. This detailed annotation process ensures that each dialogue segment within IEMOCAP is a valuable sample for emotional analysis.

The size of the corpus, along the detailed motion capture information and interactive setting that promote authentic emotions, makes this dataset a valuable addition to the existing databases for the study and modeling of multimodal and expressive human communication, demonstrated by how the dataset has been used in a variety of tasks of Emotion Recognition in many different modalities.

3.3 Data Preparation

This section gives an additional overview of the text and audio data preparation processes, adding more details in respect to what has been discussed before.

3.3.1 Text Processing

As we prepare data for SpectroBERT's training and fine-tuning, the text processing pipeline transforms raw text into a structured sequence that the model can interpret. Below are the steps:

1. **Tokenization**: The raw text is first tokenized using the BERT tokenizer from Hugging Face's Transformers [32] library. This process involves breaking down the text into a sequence of tokens and adding special tokens required by the

model, such as ‘[CLS]‘ for the start of a sequence and ‘[SEP]‘ for separation between segments.

2. **Truncation and Padding**: Sequences longer than the model's maximum sequence length are truncated to fit, while shorter sequences are padded with a special ‘[PAD]‘ token to ensure uniformity across all inputs. We use 128 as the maximum text length.

3. **Attention Masks**: An attention mask is created for each sequence, enabling the model to differentiate between the actual sequence and padded areas during self-attention calculations.

4. **Embedding Lookups**: Each token goes through an embedding layer, which converts it into an embedding vector, that encodes the semantic and syntactic information of the tokens.

3.3.2 Audio Processing

In preparing data for SpectroBERT's training and fine-tuning phases, the audio processing step transforms raw audio input into a structured format that the model can interpret. The audio processing pipeline involves the following steps:

1. **Waveform Loading**: The first step involves loading the audio files into the processing pipeline. This stage converts the raw audio files (typically in formats like WAV, although FLAC was used for WavCaps) into digital waveforms, representing the audio signal as a series of data points.

2. **Spectrogram Patch Extraction**: We use the AST Feature Extractor available on Hugging Face's Transformers library to convert the loaded audio waveforms into spectrogram patches. Each patch represents a portion of the audio signal, capturing both frequency and time information, and is then used as an input token for the model. This step implicitly includes a series of sub-steps, like normalization, feature extraction by converting the waveforms into their mel-spectrogram, and segmentation to split the spectrogram into patches.

3. **Audio Tokenization with Wav2Vec2 Layers**: The spectrogram patches are then processed through Wav2Vec2 convolutional layers to encode them into audio tokens as explained in Chapter 2.

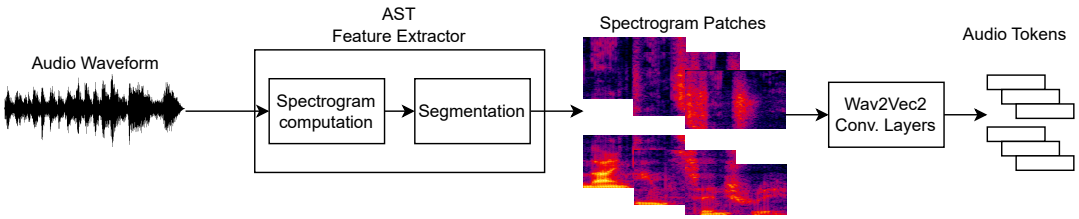

Figure 3.1: Audio Processing

Chapter 4

Experimental Settings

This chapter delves into the experimental framework designed to evaluate the efficacy of SpectroBERT, our proposed model for speech-text cross-modal learning. The experiments are suggested to test various aspects of SpectroBERT, including its ability to process and align speech and text data, its performance in comparison to existing models, and its adaptability to different types of speech and text inputs. The suggested framework not only aims to demonstrate the capabilities of SpectroBERT, but to also provide insights into the strengths and limitations of the model.

This chapter will detail the experimental setup and the metrics suggested for evaluation, while an analysis of the obtained results follows in the next chapter.

4.1 Pre-Training

In this stage of SpectroBERT's development, we employ pre-training to establish a foundational understanding of speech and text modalities. The choice of datasets, which is of fundamental importance, was previously discussed, but a summarized overview is offered below.

We suggest three distinct datasets: LibriSpeech [24], CommonVoice [25], and WavCaps [26], each offering unique characteristics and challenges.

- **LibriSpeech**: This dataset, well-regarded in speech recognition, offers a diverse range of English speech from audiobooks.
- **CommonVoice**: An open-source dataset, CommonVoice provides a wide array of accents and dialects, enhancing the diversity of the speech data.

- **WavCaps**: Unlike LibriSpeech and CommonVoice, WavCaps is not limited to speech, and it includes richly annotated textual captions that have been processed with ChatGPT, offering a structure that integrates more contextual information with the audio data.

We were able to successfully pre-train SpectroBERT on the WavCaps and LibriSpeech datasets, while CommonVoice is temporarily left aside as a future work.

4.2 Fine-Tuning

Once SpectroBERT has completed its pre-training, it enters the fine-tuning stage, where we really start to tailor the model to specialized tasks. Fine-tuning allows the model to adapt its broad understanding of speech and text to handle the specific nuances and requirements of the applications of interest. In this phase, we focus on two distinct tasks: Audio Question Answering (AQA) and Speech Emotion Recognition (SER).

- For AQA, we utilize the Clotho-AQA dataset, a challenging collection designed for evaluating a model's ability to comprehend and respond to complex audio-based questions.
- For SER, we utilize RAVDESS, and suggest to adapt it for IEMOCAP in future works, given the two datasets offer a different range of emotional speech data.

This section will explore the fine-tuning processes for these tasks, detailing the dataset-specific adaptations, training methodologies, and the learning regime that SpectroBERT undergoes to specialize in the downstream applications.

4.2.1 Evaluation Metrics

Before delving into the details of the suggested experiments, we find it convenient to introduce the metrics proposed to quantify how well a certain model is doing.

A metric can have many appearances - some might be a scalar value that has to be minimized or maximized, like a difference or score, some might be a percentage, like the proportion of correctly classified instances out of the total instances - and the choice of metrics merely depends on the nature of the task at hand.

In our case all downstream tasks are either binary or multiclass classification tasks, and for consistency with previous works, we evaluate the models in terms of

Accuracy, but we also suggest the usage of the Area Under the Receiver Operating Curve (AUROC).

Accuracy

Accuracy (Acc) is a foundational metric in machine learning, defined as the ratio of correctly predicted instances to the total number of instances in the dataset. Mathematically, it is expressed as:

$$Acc = \frac{TP + TN}{TP + TN + FP + FN}$$

where:

- TP is the number of true positives (correctly predicted positive instances).
- TN is the number of true negatives (correctly predicted negative instances).
- FP is the number of false positives (incorrectly predicted positive instances).
- FN is the number of false negatives (incorrectly predicted negative instances).

While accuracy is intuitive and provides a quick assessment of a model's overall correctness, its suitability depends on the specific characteristics of the dataset. In cases of imbalanced datasets, where one class is significantly more prevalent than others, accuracy may be misleading, and for those cases we suggest considering additional metrics such as precision, recall, and the F1 score to gain a more nuanced understanding of a model's performance.

Despite its limitations, accuracy remains a valuable metric, offering a straightforward measure of a model's effectiveness, and for consistency with previous works, we conduct our experiments only taking accuracy into account.

Area Under the Receiver Operating Curve

The Area Under the Receiver Operating Characteristic (ROC) Curve, or AUROC, is a key metric in machine learning, especially for binary classification tasks. The ROC curve shows how a model balances between correctly predicting positive cases (true positive rate or sensitivity) and incorrectly predicting negative cases as positive (false positive rate, which is 1 minus specificity).

In other words, it measures how well a model can differentiate between two classes across all possible thresholds, with values ranging from 0 to 1, where a score of 0.5 indicates random chance, and a score of 1 suggests perfect discrimination. Mathematically, AUC-ROC is calculated as the area under the ROC curve:

$$AUC\text{-}ROC = \int_0^1 \mathrm{TPR}(\mathrm{FPR}^{-1}(t))\, dt$$

where TPR is the true positive rate (sensitivity), FPR is the false positive rate, and $\mathrm{FPR}^{-1}(t)$ is the inverse function of FPR.

A higher AUROC value means the model does a better job at telling positives from negatives. Here's a general interpretation of AUROC values:

- **0.5**: This indicates a model that performs no better than random chance. The curve hugs the diagonal line, and the model is not effectively distinguishing between the two classes.
- **0.7-0.8**: This is considered acceptable. The model has some discriminatory power, but there is room for improvement.
- **0.8-0.9**: This is considered good. The model has a strong discriminatory power and is effective in distinguishing between the positive and negative classes.
- **0.9 and above**: This is considered excellent. The model has a high discriminatory power, and its ability to differentiate between classes is very strong.

AUROC becomes particularly valuable in scenarios where the class distribution is imbalanced. However, it is essential to consider the context of the problem and the specific consequences of false positives and false negatives, as AUC-ROC alone may not provide a complete picture of a model's performance.

4.2.2 Parameters Freezing

When fine-tuning a complex model like SpectroBERT, we often use a technique called 'parameters freezing' or weight freezing. This method involves selectively stopping updates to certain parts of the model's weights to preserve what the model learned during pre-training while allowing it to adjust to new tasks during fine-tuning. In fine-tuning SpectroBERT, we want to see how freezing some layers of the model impacts the training behaviour and downstream performance.

Good candidates for freezing are the initial layers, which capture more universal aspects of language and audio, and don't always need to be adjusted for specific tasks. They've already learned a lot about language and sound during pre-training, so by freezing their parameters, we make sure SpectroBERT retains its foundational understanding of these basic elements, while the layers closer to the end of the

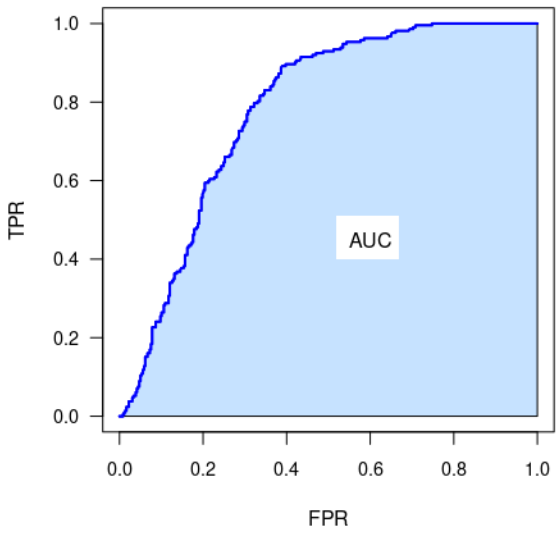

Figure 4.1: ROC and Area Under the Receiver Operating Curve. Credits to the original author [33].

model - those that directly deal with the task-specific data - are fine-tuned, helping these parts of the model become more skilled at handling the specific requirements of the tasks we're focusing on.

Deciding which parameters to freeze depends on various factors, like how big and representative the fine-tuning dataset is, the complexity of the task, and how similar the fine-tuning task is to what the model learned during pre-training. We can think of this process in mathematical terms as well:

$$\theta_{\text{frozen}} = \text{constant}$$

$$\theta_{\text{fine-tune}} = \arg\min_{\theta} L(\theta, \mathcal{D}_{\text{fine-tune}})$$

In this representation, θ_{frozen} are the weights that stay the same during fine-tuning, and $\theta_{\text{fine-tune}}$ are the ones we update. L is the loss function we're trying to minimize over the fine-tuning dataset $\mathcal{D}_{\text{fine-tune}}$. This approach helps balance preserving the general knowledge SpectroBERT has gained and adapting it to new, specific tasks.

4.2.3 Audio Question Answering

The capability of SpectroBERT to comprehend and integrate complex audio-text relationships are put to the test in the Audio Question Answering (AQA) task.

We suggest a series of experiments to assess how different pre-training strategies and weight freezing configurations affect the model's performance in both 'binary' (yes/no) and 'open' (single-word) AQA tasks.

For each variant of the AQA task, we propose four main experiments:

- Without Pre-Training: The model is initialized with BERT's [3] pretrained weights (for the transformer stack and the embedding layers) and random weights (for everything else), and directly fine-tuned on the AQA tasks, serving as a baseline to understand the importance of pre-training.
- Pre-Trained on Each Dataset: Here, the model is pre-trained separately on LibriSpeech, Common Voice, and WavCaps, and then fine-tuned for the AQA tasks, to test the impact of domain-specific pre-training on AQA performance.
- Full Model Fine-Tuning: The model, pre-trained on the above datasets, is fine-tuned with all weights unfrozen, allowing the entire model to adapt to the AQA tasks.
- Partial Model Fine-Tuning: all layers except the classification head (and in another variant, also the last layer of the transformer) are frozen, to determine whether preserving lower-level representations while adapting only the higher-level representations is beneficial, or excessively restricting.

For each of these configurations, the model is fine-tuned on the Clotho-AQA dataset, separately for binary and open question-answer pairs.

We propose to measure the model's performance using standard classification metrics such as accuracy for binary questions and also top-5 and top-10 accuracy for open questions. These metrics provides insights into how well the model can generalize from its training data to accurately answer unseen questions.

The results from these experiments are expected to reveal the importance of pre-training in AQA tasks, the effectiveness of domain-specific knowledge, and the impact of freezing various parts of the model during fine-tuning.

4.2.4 Speech Emotion Recognition

The Speech Emotion Recognition (SER) experiments for SpectroBERT aimed to understand how the model interprets and classifies emotional nuances within speech. SER is a task that requires high sensitivity to acoustic variations, and SpectroBERT's ability to integrate audio and text data is crucial for its performance.

We followed a similar experimental structure to the AQA tasks to evaluate SpectroBERT's capabilities in this domain:

- Without Pre-Training: Like for AQA, SpectroBERT is assessed without any pre-training, using BERT's pretrained weights (for the transformer stack and the embedding layers) and random weights (for everything else). This experiment establishes a baseline for the model's intrinsic SER capabilities without the influence of transfer learning.
- Pre-Trained on Each Dataset: Here, the model is pre-trained separately on LibriSpeech, Common Voice, and WavCaps, and then fine-tuned for the SER tasks to test the impact of domain-specific pre-training on SER performance.
- Full Model Fine-Tuning: After pre-training, the entire model undergoes fine-tuning on the RAVDESS and IEMOCAP datasets with all layers unfrozen, to test the model's ability to adapt its entire range of learned parameters to the SER task.
- Partial Model Fine-Tuning: To understand the benefits of focused adaptation, certain layers of SpectroBERT are frozen, leaving only the classification head (and in some variants, the last layer of the transformer too) unfrozen. This allows to investigate whether maintaining the pre-trained features in earlier layers while fine-tuning deeper layers would yield better SER performance.

The RAVDESS and IEMOCAP datasets offer a rich array of emotional expressions, providing a solid testing ground for SpectroBERT. We were able to successfully fine-tune SpectroBERT on the RAVDESS dataset, with its acted speech that present clear emotional categories, while IEMOCAP, with its more natural dialogues, is temporarily left aside as a future work.

For evaluation, we propose using accuracy at top-1 and top-2 only, given the low amount of classes.

Chapter 5

Experimental Result and Analysis

In this chapter, we are going to analyse how well SpectroBERT did during its training phases, in order to figure out what the model is good at and where it might need some work. We will start by looking at the pre-training phase, with our main focus being on the differences between the pre-training results when using WavCaps and LibriSpeech, and how each of these affected the model's learning behaviour.

After that, we will move on to how SpectroBERT performed when fine-tuned for specific tasks. We will cover its performance in Audio Question Answering (AQA) tasks, both when the answers were just 'yes' or 'no' and when they were more open-ended, and we will also look at how it did in recognizing emotions in speech. This part will give us a good idea of how flexible and accurate the proposed framework is when it comes to dealing with different kinds of challenges.

By the end of this chapter, we aim to have a solid understanding of SpectroBERT's current capabilities and a few ideas on where to take it next, while in the next (and final) Chapter 6 we will pull together what has been learned from these analyses, to obtain a clear picture of where SpectroBERT stands right now, how it performed in our tests, and what could be improved in the future.

5.1 Pre-Training

5.1.1 Differences between WavCaps and LibriSpeech

In this section, we analyse how SpectroBERT's training differed when using two distinct datasets: WavCaps [26] and LibriSpeech [24]. Our findings in Figure 5.1

indicate that using WavCaps generally led to better training outcomes, as reflected by lower loss values during the training process, suggesting that the type of data SpectroBERT is trained on can significantly impact its learning efficiency and effectiveness.

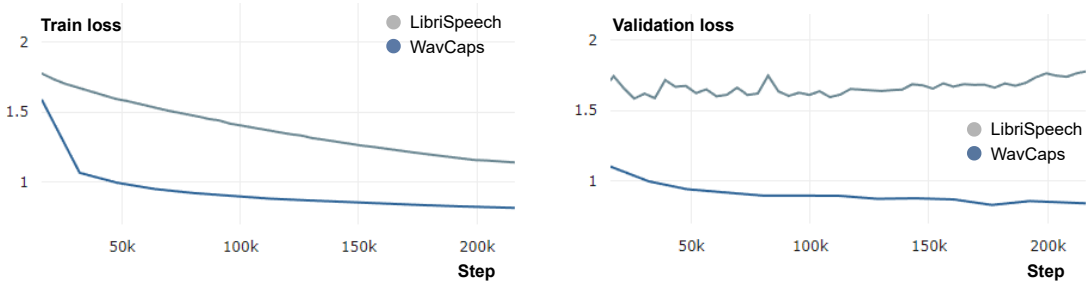

Figure 5.1: Pre-training graphs.

WavCaps seems to have provided a richer training ground for SpectroBERT, with its diverse and complex audio environments that might have helped the model learn more robust and adaptable representations of speech, as evidenced by the generally-lower and smoother loss curves. It's possible that the broader range of acoustics and contexts in WavCaps challenged the model in ways that better prepared it for real-world applications. On the other hand, LibriSpeech, while still a valuable dataset, appears to have offered a somewhat limited learning experience, perhaps because its clean audiobook-style speech, may not have presented the same level of complexity and diversity as WavCaps. This could explain why the training loss with LibriSpeech was generally higher compared to WavCaps.

The comparison of these two datasets in training SpectroBERT highlights the importance of dataset selection in Machine Learning and how varying audio characteristics can influence a model's learning behaviour.

In the following experiments, we will only be using WavCaps as a pre-training dataset, given its prospected better performance.

5.2 Fine-Tuning

5.2.1 AQA Binary

In this section, we explore how SpectroBERT performed on the Audio Question Answering (AQA) task with binary (yes/no) responses. Specifically, we looked at results from a setup without pre-training and several setups with pre-training on the WavCaps dataset, but with varying degrees of parameters freezing.

Firstly, we have the scenario where SpectroBERT was fine-tuned without any pre-training. This serves as a baseline to understand the importance of pre-training in preparing the model for specific tasks. Then, in the pre-trained setups we experimented with freezing different layers:

- **Freezing everything (bin-frz).** In this experiment, we froze the entire network except the classification head. This setup is similar to what is commonly done when fine-tuning a model.
- **Freezing everything except last Transformer layer (bin-frz-nolast).** In this experiment, starting from the setup of the previous one, we unfroze the very last layer of the Transformer stack, in an attempt to let it learn the interaction of audio and text specific for this task.
- **Freezing everything except the audio encoder (bin-frz-noaudio).** In this experiment , we moved in a different direction. While keeping the entire Transformer stack frozen as in the first experiment, we unfroze the Wav2Vec2-inspired [16] convolutional layers that extract audio features. The idea here was to let those layers learn to extract audio clues that might be specific for the task.

Figure 5.2 shows the different combinations of freezing. These varying degrees of freezing allows us to observe how restricting updates to certain parts of the model impacts its ability to handle the AQA task. Generally, freezing layers is thought to help retain the general knowledge gained during pre-training, but it also limits the model's ability to adapt to the specifics of a new task. The results from these experiments can better help understand how to balance between preserving pre-learned knowledge and adapting to new tasks.

In figure 5.3 the losses behaviour, together with the accuracy values, are reported.

Our findings show that all the three pre-trained versions of SpectroBERT outperformed the non-pretrained setup, highlighting the importance of pre-training in preparing the model for specific tasks. However, we also observed that setups with less freezing generally performed better, indicating a benefit in allowing more layers to adapt to the task. This improved performance, nonetheless, came with a caveat: a tendency to overfit more quickly.

Overfitting is a common issue in machine learning where a model learns the details and noise in the training data to an extent that it negatively impacts the performance of the model on new data. Essentially, an overfitted model is too well-tuned to the training data: it's great at handling that specific set of data, but it doesn't generalize well to other, unseen datasets.

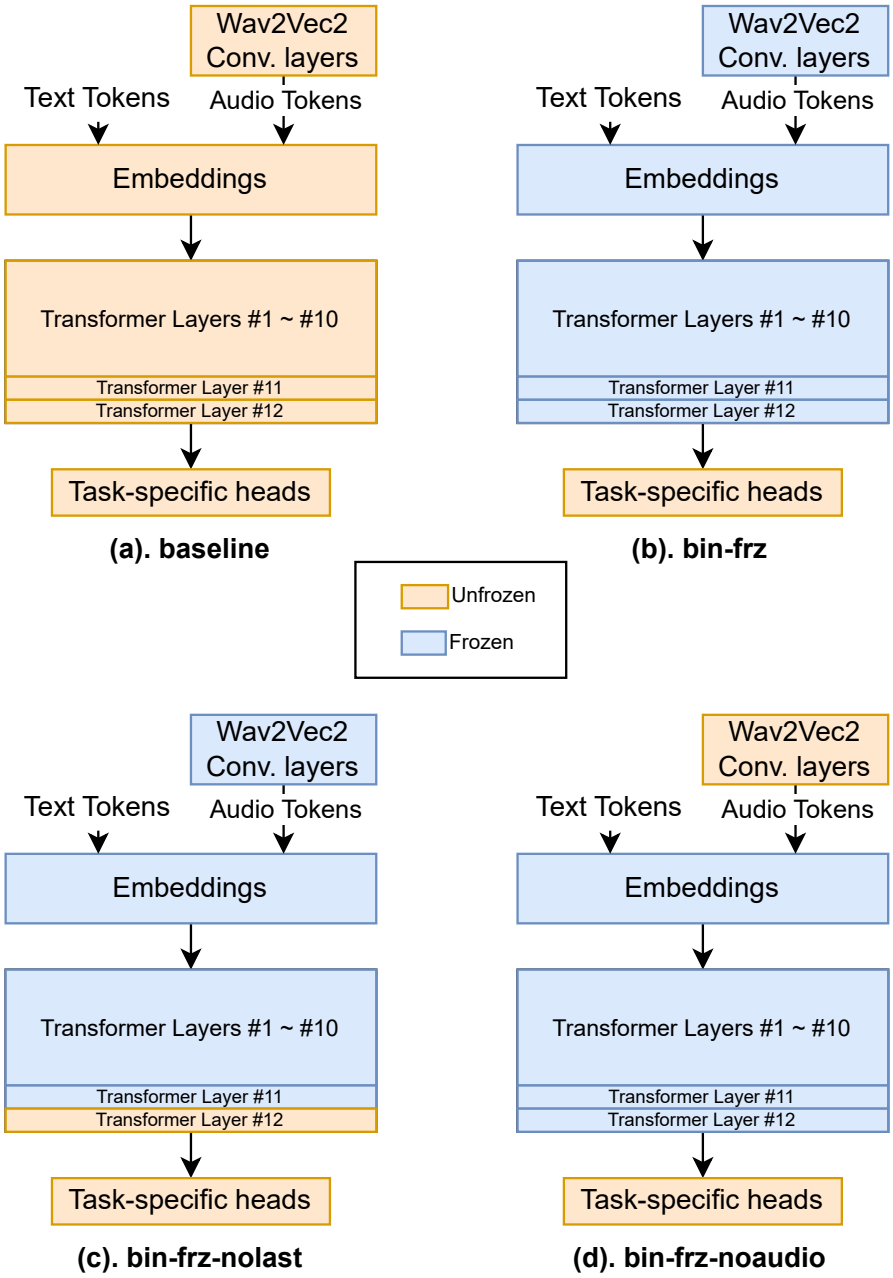

Figure 5.2: Different combinations of freezing in the AQA-binary setting

This happens when the model becomes complex enough to capture not just the underlying patterns in the training data, but also the random fluctuations and irrelevant features (noise). As a result, while the model might perform exceptionally well on the training data, showing high accuracy and low loss, its performance

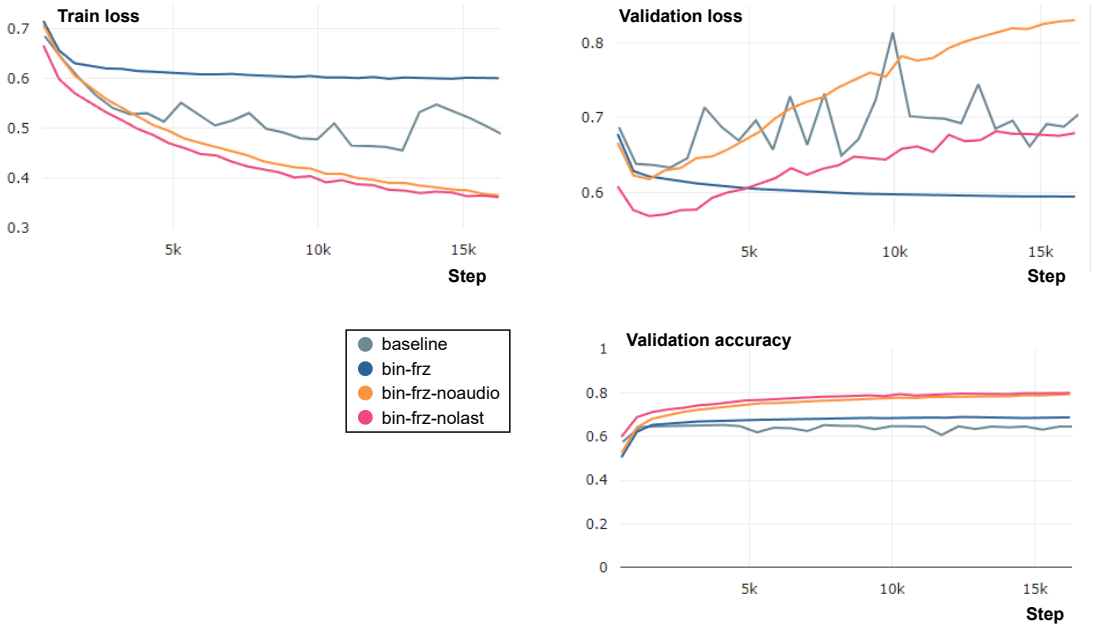

Figure 5.3: AQA-binary training graphs.

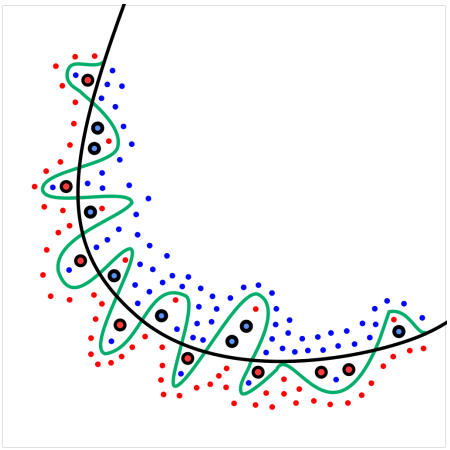

Figure 5.4: Overfitting visualized. The green line depicts an overfitted model, closely tracing the training data but likely less effective on new, unseen data (shown as black-outlined dots). In contrast, the black line represents a regularized model, balancing data fit and generalization, potentially performing better on new data. Credits to the original author [34].

drops significantly when presented with new, unseen data, as visualized in Figure 5.4.

Nevertheless, if we consider the results at a step where the validation loss has not diverged too much from the training loss - which is a common signal of overfitting - we find that the versions of SpectroBERT with a balanced degree of freezing outperformed the version where the entire network is frozen, as shown in Figure 5.5.

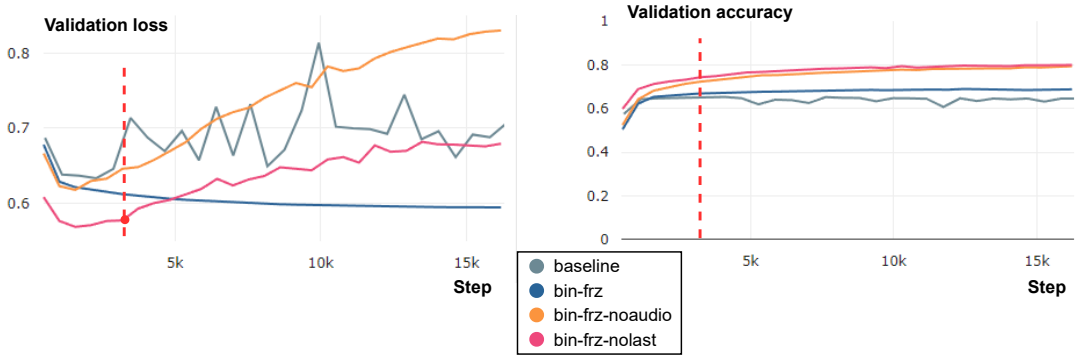

Figure 5.5: Comparison between varying degrees of freezing at early steps.

With these considerations in mind, Table 5.1 compares in terms of accuracy our results - cherry-picked at a early step of the training to avoid overfitting - with the results of Clotho-AQA's original paper.

Table 5.1: Accuracies (%) of 'yes' or 'no' binary classifier on Clotho-AQA

Experiment	Pre-training	Epoch	Top-1 Acc.
ClothoAQA	-	-	62.7
baseline	None	4	64.9
bin-frz	WavCaps	30	68.8
bin-frz-nolast	WavCaps	6	**74.3**
bin-frz-noaudio	WavCaps	4	69.6

5.2.2 AQA Open

In this section, we analyze SpectroBERT's performance on the open-ended Audio Question Answering (AQA) task under various experimental setups. These experiments were designed to assess how different levels of pre-training and parameter freezing impact the model's ability to handle more complex, open-ended questions.

Similar to the AQA-binary setup, we first have the scenario where SpectroBERT was fine-tuned without any pre-training. This setup serves as a baseline, helping us understand the importance of pre-training in preparing the model for complex questioning scenarios.

Then, in the pre-trained setups we experimented with freezing different layers:

- **No freezing (open-nofrz).** In this experiment, no freezing was applied during fine-tuning. This allowed all layers of the model to adjust freely to the specifics of the open AQA task, potentially enhancing task-specific learning.
- **Freezing everything except last Transformer layer (open-frz-nolast).** In this experiment, everything was frozen except the classification head and the last layer of the Transformer stack. The idea was to allow the final layer to adapt to the intricacies of the task while retaining the previously learned knowledge in the other layers.
- **Freezing everything except last 2 Transformer layer (open-frz-nolast2).** In this experiment, expanding on the previous setup, in this experiment we unfroze the last two layers of the Transformer stack along with the classification head. This provides even more flexibility for the model to adapt to the AQA task, potentially capturing more complex interactions between audio and text necessary for open-ended questions.

Figure 5.6 shows the different combinations of freezing. We analyse how these varying degrees of freezing impacts the model ability to handle the AQA-open task. In figure 5.7 the losses behaviour, together with the accuracy values, are reported.

Like in the binary case, our findings show that all the three pre-trained versions of SpectroBERT outperformed the non-pretrained setup, highlighting the importance of pre-training in preparing the model for specific tasks, but differently from the binary case, overfitting does not seem to be too much of a problem in this experiments. We attribute the cause of this finding to the more complex nature of the open-ended single-word question answering compared to the simpler 'yes/no' scenario.

Table 5.2 compares in terms of accuracy our results with the ones of Clotho-AQA's original paper [27].

5.2.3 SER

In this section, we delve into SpectroBERT's performance in the Speech Emotion Recognition (SER) task under different training and fine-tuning conditions. Our experiments aimed to understand how the model reacts to varying levels of pre-training and parameter freezing, especially in a task that focuses more on emotional

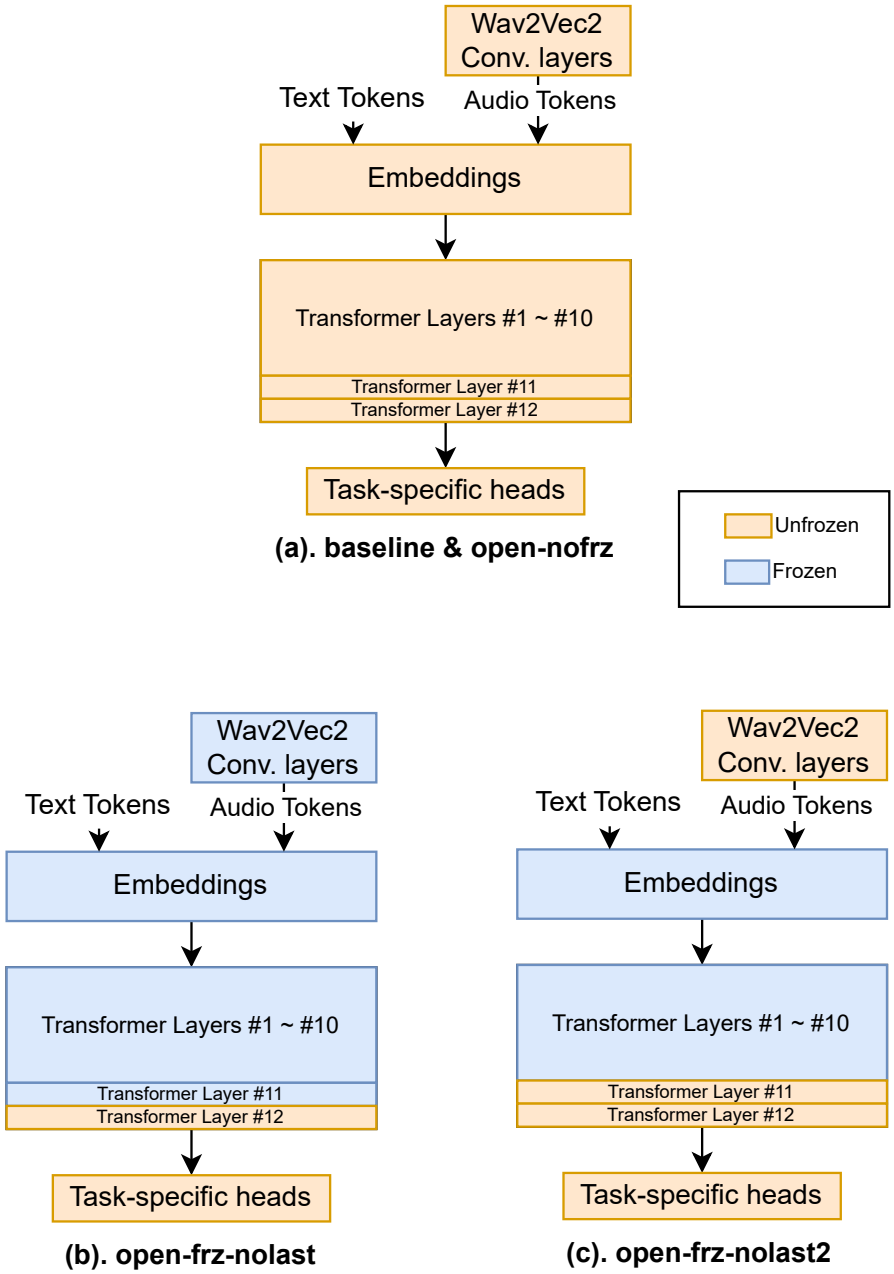

Figure 5.6: Different combinations of freezing in the AQA-open setting

nuances rather than content.

Similar to the AQA tasks, we initially fine-tuned SpectroBERT on the RAVDESS dataset without any pre-training. This setup helps us evaluate the model's innate

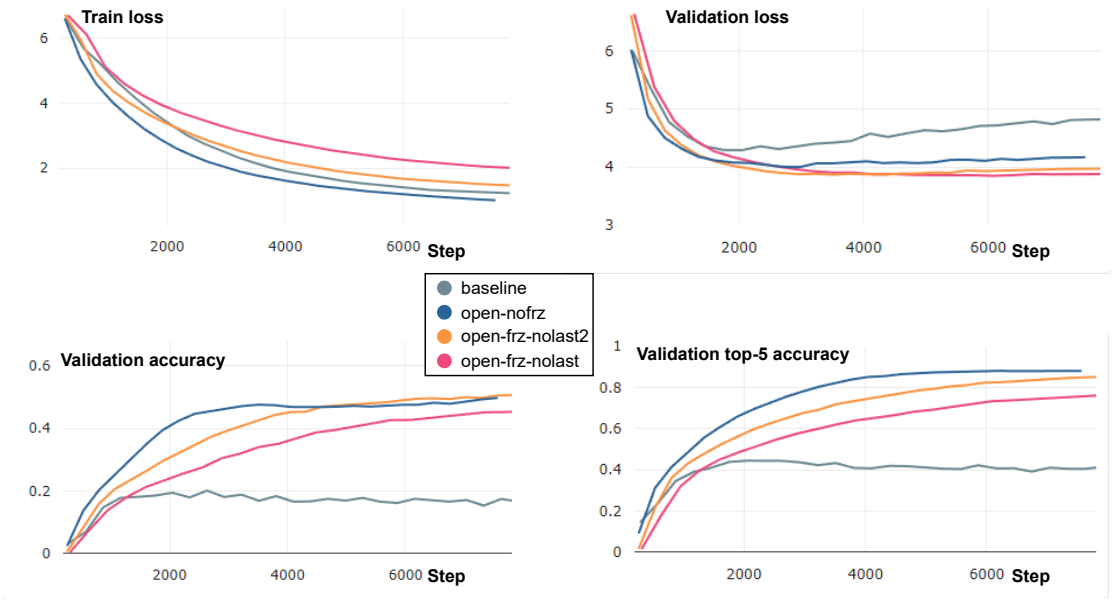

Figure 5.7: AQA-open training graphs.

Table 5.2: Accuracies (%) of single-word answers multi class classifier on ClothoAQA

Experiment	Pre-training	Epoch	Top-1 Acc.	Top-5 Acc.
Clotho-AQA	-	-	**54.2**	**93.7**
baseline	None	9	20.1	44.4
open-nofrz	WavCaps	28	49.7	**87.9**
open-frz-nolast	WavCaps	30	46.7	78.3
open-frz-nolast2	WavCaps	30	**50.8**	85.4

ability to recognize emotions without the benefit of prior learning, and provides a baseline.

Then, in the WavCaps pre-trained setups we experimented with freezing different layers:

- **Freezing everything except last Transformer layer (ser-frz-nolast).** In this experiment, fine-tuning is performed freezing all but the classification head and the last layer of the Transformer stack. This setup allowed us to see how much the final Transformer layer contributes to understanding emotions when most of the model's parameters are fixed.
- **Freezing everything except the audio encoder (ser-frz-nolast-noaudio).**

In this experiment, expanding on the previous configuration, we unfreeze not only the last Transformer layer and the classification head, but also the audio feature extractor (Wav2Vec2 convolutional layers). This was done to assess whether allowing these layers to adapt would enhance the model's ability to extract emotion-relevant information from the audio.

Figure 5.8 shows the different combinations of freezing, while in Figure 5.9 the losses behaviour, together with the accuracy values, are reported.

Interestingly, these results indicate that the difference between pre-trained and non-pre-trained versions of SpectroBERT for SER is not as pronounced as one might expect, suggesting that the task of emotion recognition may rely less on content understanding and more on picking up emotional cues, an ability on which pre-training might not significantly impact.

Instead, a notable improvement was observed in experiment #2, when the audio feature extractor was unfrozen. This implies that allowing these layers to adjust and learn specifically from emotion-related audio features is beneficial for the task, suggesting that in this case the emotional tone of the audio is more crucial than the linguistic content itself.

Table 5.3 compares in terms of accuracy the results obtained in the various experiments.

Table 5.3: Accuracies (%) of emotional multi class classifier on RAVDESS

Experiment	Pre-training	Epoch	Top-1 Acc.	Top-2 Acc.
baseline	None	47	42.3	67.3
ser-frz-nolast	WavCaps	28	29.9	50.4
ser-frz-nolast-noaudio	WavCaps	23	**63.2**	**82.1**

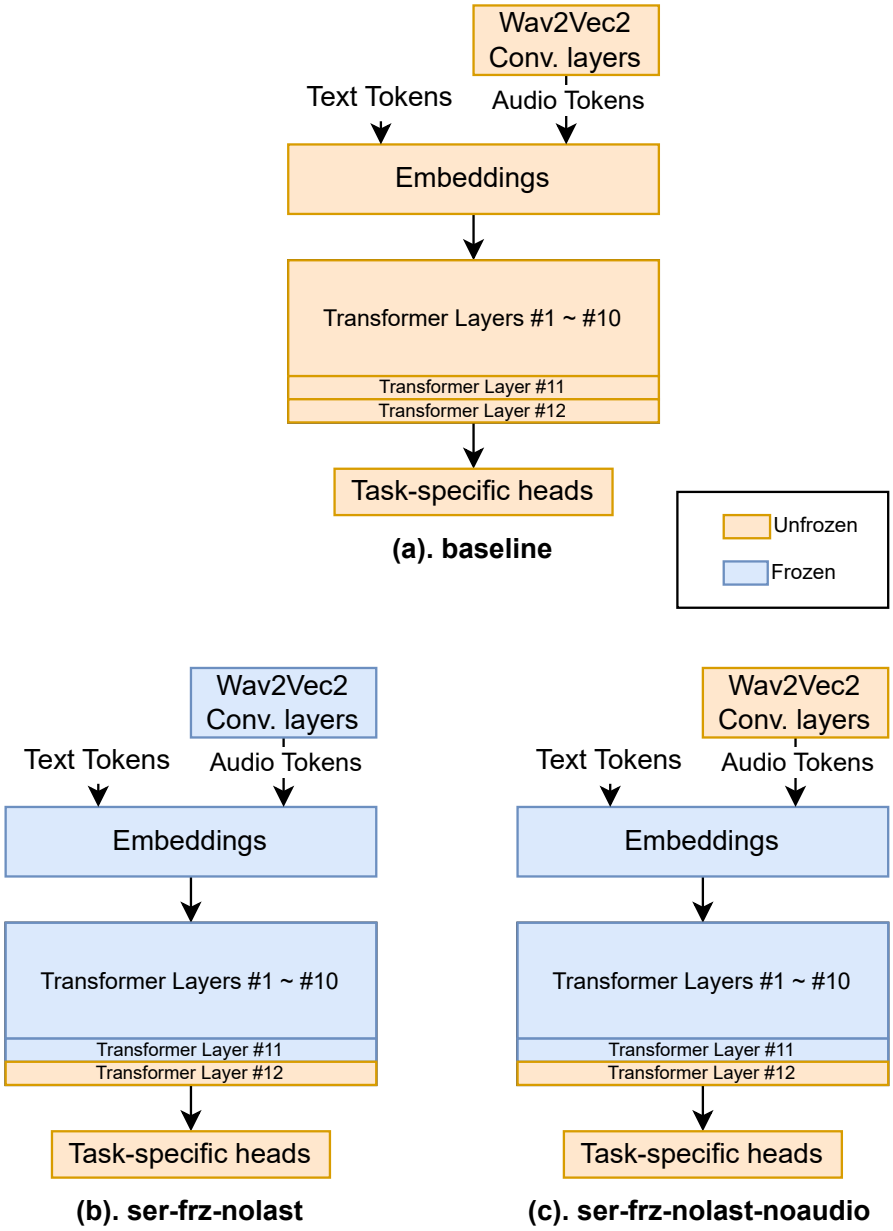

Figure 5.8: Different combinations of freezing

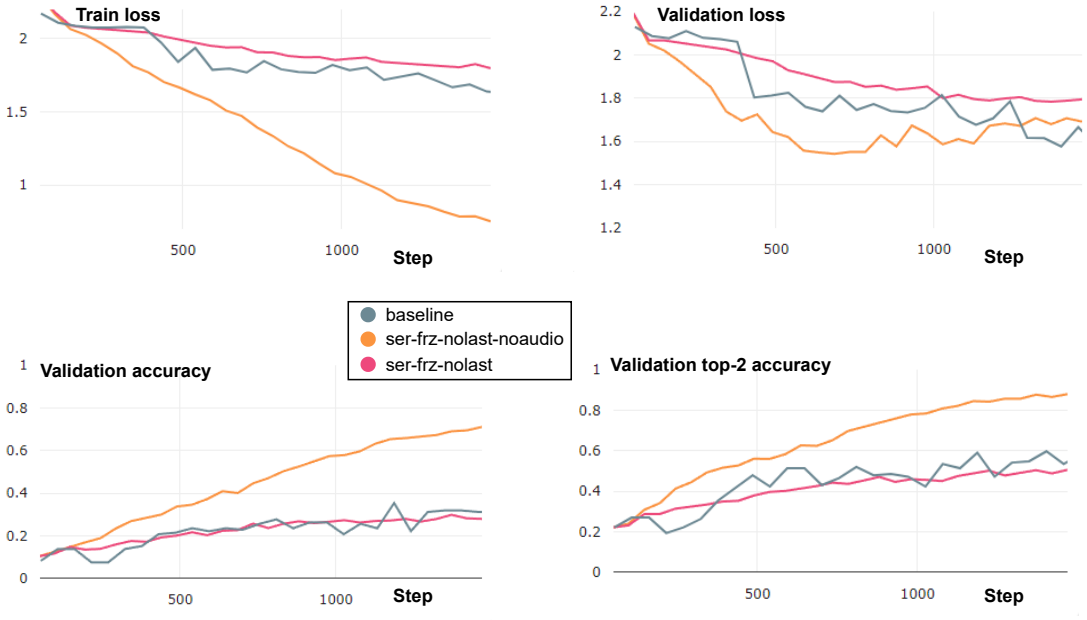

Figure 5.9: SER graphs.

Chapter 6

Conclusion

6.1 Final Considerations

As we reach the conclusion of this thesis, several key takeaways and directions for future works emerge from the analysis of SpectroBERT.

- First and foremost, our experiments and the results obtained throughout the development of SpectroBERT offer a promising validation of the fundamental premise of this thesis: it is indeed possible to create a model that can seamlessly integrate and interpret speech and text data by leveraging self-attention; this approach, previously proven effective in VisualBERT for bridging visual and textual data, is successfully adapted to connect speech and text modalities instead, demonstrating once again the versatility and potential of Transformer-based models in multimodal contexts.
- Second, another key insight from our experiments is the effectiveness of the two-step process of pre-training and fine-tuning. Our results have shown that pre-training SpectroBERT on diverse datasets, like we did with WavCaps, can significantly contributes to its performance in downstream tasks, as it was the case in the Audio Question Answering task. On the other hand, fine-tuning after pre-traning might only give a slight advantage over training from scratch if the pre-training dataset is not aligned in nature and distribution of data with the fine-tuning one, as it was the case in the Speech Emotion Recognition. As such, we suggest that pre-training a model in the multimodal context be an essential part of future works in the field, but with a careful choice of pre-training datasets.
- Third, is the careful approach to choosing a strategy of parameter freezing during fine-tuning. Our results suggest that carefully choosing which layers to freeze or unfreeze can greatly impact the model's performance on specific tasks.

As seen in tasks like Speech Emotion Recognition - where allowing the model to adapt its audio feature extraction layers led to noticeable improvements - this finding points to the potential benefits of layer-specific fine-tuning strategies depending on the nature of the task at hand.

In conclusion, we are confident that the main goal of researching and developing a framework for *"Speech-Text Cross-Modal Learning through Self-Attention Mechanisms"* - this thesis' title - has been accomplished, by providing the reader with an introductory background of the problem at hand first, and by having later proposed a novel architecture which we named SpectroBERT, which has been demonstrated to work through our suite of experiments.

6.2 Future works

Looking ahead, there are several possibilities for future research and development of our proposed method.

One immediate area is the exploration of the tasks and configurations that could not be tested in the current phase of our work, and thoroughly experiment with all the different combinations of datasets - like Common Voice and IEMOCAP, which have been left aside in this phase of our work - and fine-tuning strategies - like different layer-specific freezing strategies - to further study how the choice of the pre-training datasets, and the varying degree of freezing, impacts on the downstream tasks' performance.

Another promising direction is the exploration of SpectroBERT's applicability in more diverse and challenging real-world scenarios, extending its use beyond the tasks of Audio Question Answering and Speech Emotion Recognition.

For example, investigating its performance using different languages and dialects, or in environments with varying levels of background noise, could provide valuable information about its practical utility and limitations. Also, an extension to the aforementioned sequence-to-sequence class of tasks, or to the generative class of tasks, is to be taken into consideration.

Bibliography

[1] Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. «Visualbert: A simple and performant baseline for vision and language». In: *arXiv preprint arXiv:1908.03557* (2019) (cit. on pp. i, 1, 2, 19, 22, 24, 30).

[2] Yuan Gong, Yu-An Chung, and James Glass. «Ast: Audio spectrogram transformer». In: *arXiv preprint arXiv:2104.01778* (2021) (cit. on pp. i, 1, 2, 20, 24).

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. «Bert: Pre-training of deep bidirectional transformers for language understanding». In: *arXiv preprint arXiv:1810.04805* (2018) (cit. on pp. i, 16, 18, 21, 22, 25, 28, 30, 52).

[4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. «Attention is all you need». In: *Advances in neural information processing systems* 30 (2017) (cit. on pp. i, 15, 17, 19, 20, 24, 26, 28, 35).

[5] A. M. Turing. «Computing Machinery and Intelligence». In: *Mind* 59.236 (1950), pp. 433–460. ISSN: 00264423, 14602113. URL: http://www.jstor.org/stable/2251299 (cit. on p. 4).

[6] Microsoft Corporation. *Bing AI Image Generator*. https://www.bing.com/. Accessed: 29/11/2023. 2023 (cit. on p. 5).

[7] *Data Mining Vs Machine Learning Vs Artificial Intelligence*. https://www.softwaretestinghelp.com/data-mining-vs-machine-learning-vs-ai/. Accessed: 29/11/2023 (cit. on p. 9).

[8] Brian Mwandau. *Investigating Keystroke Dynamics as a Two-Factor Biometric Security*. https://www.researchgate.net/publication/325870973_Investigating_Keystroke_Dynamics_as_a_Two-Factor_Biometric_Security. Accessed: 29/11/2023. 2018 (cit. on p. 10).

[9] Swapna K E. *Convolution Neural Network - Deep Learning*. https://developersbreach.com/convolution-neural-network-deep-learning/. Accessed: 2023-11-29 (cit. on p. 11).

[10] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. «Rich feature hierarchies for accurate object detection and semantic segmentation». In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 580–587 (cit. on pp. 11, 12).

[11] Corinna Cortes and Vladimir Vapnik. «Support-vector networks». In: *Machine learning* 20 (1995), pp. 273–297 (cit. on p. 12).

[12] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. «Faster r-cnn: Towards real-time object detection with region proposal networks». In: *Advances in neural information processing systems* 28 (2015) (cit. on pp. 12, 13).

[13] Ronald J Williams and David Zipser. «A learning algorithm for continually running fully recurrent neural networks». In: *Neural computation* 1.2 (1989), pp. 270–280 (cit. on p. 14).

[14] Sepp Hochreiter and Jürgen Schmidhuber. «Long Short-term Memory». In: *Neural computation* 9 (Dec. 1997), pp. 1735–80. DOI: 10.1162/neco.1997.9.8.1735 (cit. on p. 14).

[15] Ashutosh Tripathi NLP. *What is the main difference between RNN and LSTM - NLP RNN vs LSTM*. https://ashutoshtripathi.com/2021/07/02/what-is-the-main-difference-between-rnn-and-lstm-nlp-rnn-vs-lstm/. Accessed: 2023-11-29. 2021 (cit. on p. 15).

[16] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. «wav2vec 2.0: A framework for self-supervised learning of speech representations». In: *Advances in neural information processing systems* 33 (2020), pp. 12449–12460 (cit. on pp. 21, 25, 26, 56).

[17] Steffen Schneider, Alexei Baevski, Ronan Collobert, and Michael Auli. «wav2vec: Unsupervised pre-training for speech recognition». In: *arXiv preprint arXiv:1904.05862* (2019) (cit. on p. 21).

[18] Kaicheng Yang, Hua Xu, and Kai Gao. «CM-BERT: Cross-Modal BERT for Text-Audio Sentiment Analysis». In: *Proceedings of the 28th ACM International Conference on Multimedia*. MM '20. Seattle, WA, USA: Association for Computing Machinery, 2020, pp. 521–528. ISBN: 9781450379885. DOI: 10.1145/3394171.3413690. URL: https://doi.org/10.1145/3394171.3413690 (cit. on p. 22).

[19] Yung-Sung Chuang, Chi-Liang Liu, Hung-Yi Lee, and Lin-shan Lee. «Speechbert: An audio-and-text jointly learned language model for end-to-end spoken question answering». In: *arXiv preprint arXiv:1910.11559* (2019) (cit. on p. 22).

[20] Hang Li, Yu Kang, Tianqiao Liu, Wenbiao Ding, and Zitao Liu. «CTAL: Pre-training cross-modal transformer for audio-and-language representations». In: *arXiv preprint arXiv:2109.00181* (2021) (cit. on pp. 22, 23).

[21] Alan V Oppenheim. *Discrete-time signal processing*. Pearson Education India, 1999 (cit. on p. 26).

[22] Petre Stoica, Randolph L Moses, et al. *Spectral analysis of signals*. Vol. 452. Pearson Prentice Hall Upper Saddle River, NJ, 2005 (cit. on p. 26).

[23] Rico Sennrich, Barry Haddow, and Alexandra Birch. «Neural machine translation of rare words with subword units». In: *arXiv preprint arXiv:1508.07909* (2015) (cit. on p. 27).

[24] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. «Librispeech: an asr corpus based on public domain audio books». In: *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE. 2015, pp. 5206–5210 (cit. on pp. 34, 40, 47, 54, 76).

[25] Rosana Ardila et al. «Common voice: A massively-multilingual speech corpus». In: *arXiv preprint arXiv:1912.06670* (2019) (cit. on pp. 34, 41, 47, 76).

[26] Xinhao Mei, Chutong Meng, Haohe Liu, Qiuqiang Kong, Tom Ko, Chengqi Zhao, Mark D Plumbley, Yuexian Zou, and Wenwu Wang. «Wavcaps: A chatgpt-assisted weakly-labelled audio captioning dataset for audio-language multimodal research». In: *arXiv preprint arXiv:2303.17395* (2023) (cit. on pp. 34, 42, 47, 54).

[27] Konstantinos Drossos, Samuel Lipping, and Tuomas Virtanen. «Clotho: An audio captioning dataset». In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2020, pp. 736–740 (cit. on pp. 36, 42, 60).

[28] Shashidhar G Koolagudi and K Sreenivasa Rao. «Emotion recognition from speech: a review». In: *International journal of speech technology* 15 (2012), pp. 99–117 (cit. on p. 36).

[29] Steven R Livingstone and Frank A Russo. «The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS): A dynamic, multimodal set of facial and vocal expressions in North American English». In: *PloS one* 13.5 (2018), e0196391 (cit. on pp. 38, 43).

[30] Carlos Busso, Murtaza Bulut, Chi-Chun Lee, Abe Kazemzadeh, Emily Mower, Samuel Kim, Jeannette N Chang, Sungbok Lee, and Shrikanth S Narayanan. «IEMOCAP: Interactive emotional dyadic motion capture database». In: *Language resources and evaluation* 42 (2008), pp. 335–359 (cit. on pp. 38, 44).

[31] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. «Sequence to sequence learning with neural networks». In: *Advances in neural information processing systems* 27 (2014) (cit. on p. 39).

[32] *Transformers: State-of-the-Art Natural Language Processing*. Accessed: 29/11/2023. URL: https://huggingface.co/transformers/ (cit. on p. 44).

[33] *Partial Area Under the ROC Curve*. https://en.wikipedia.org/wiki/Partial_Area_Under_the_ROC_Curve. Accessed: 2023-11-29 (cit. on p. 51).

[34] *Overfitting*. https://en.wikipedia.org/wiki/Overfitting. Accessed: 2023-11-29 (cit. on p. 58).

[35] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. «On the variance of the adaptive learning rate and beyond». In: *arXiv preprint arXiv:1908.03265* (2019) (cit. on p. 72).

[36] Diederik P Kingma and Jimmy Ba. «Adam: A method for stochastic optimization». In: *arXiv preprint arXiv:1412.6980* (2014) (cit. on p. 72).

[37] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. «Dropout: A Simple Way to Prevent Neural Networks from Overfitting». In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html (cit. on p. 73).

[38] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. «On the difficulty of training recurrent neural networks». In: *International conference on machine learning*. Pmlr. 2013, pp. 1310–1318 (cit. on p. 73).

[39] Hugging Face. *Hugging Face Datasets*. https://huggingface.co/datasets. Accessed: 2023-11-29. 2023 (cit. on p. 76).

[40] Python Software Foundation. *Python Programming Language - Official Website*. https://www.python.org/. Accessed: 2023-11-29. 2023 (cit. on p. 79).

Appendix A

Training settings

In developing SpectroBERT, we made a series of choices regarding our training settings which we share for those interested in replicating our experiments or gaining deeper insights into our methodology.

Hardware Specifications
All of our experiments, in both the pre-training and fine-tuning stages, were conducted on a single NVIDIA V100 GPU, equipped with 32 GB of VRAM. This choice of hardware provided the necessary computational power and memory capacity to handle the complexity and size of SpectroBERT.

Batch Size
In all of our experiments we set the batch sizes to either 24 or 26, which closely met the limitations of our hardware depending on the task at hand.

Optimization
For optimization, we employed RAdam [35] (Rectified Adam), an optimizer known for its refinement over the standard Adam [36] algorithm. RAdam introduces a term that rectifies the variance of the adaptive learning rate, ensuring more stable and consistent optimization, especially in the early stages of training. This rectification improves convergence and offers better performance with fewer hyperparameter tuning requirements compared to traditional Adam. We set the initial learning rate to 1e-5, aiming for stability throughout the training process.

Training Duration
Our models were trained with maximum epochs either 30 or 50, depending on the task at hand. During pre-training for example, the maximum epochs were set to 50 to give the model enough time to truly learn from the data, while during fine-tuning on smaller datasets, the maximum epochs were set to 30.

Regularization Techniques

To further guard against overfitting and to enhance the model's generalization capabilities, we incorporated dropout layers [37] within the model, with dropout probability set to 0.1. These layers randomly deactivate certain neurons during training, which helps the model to learn more robust features.

Additionally, we used gradient clipping [38], a technique that prevents the exploding gradient problem by clipping them to a maximum value - 5.0 in our case - thereby ensuring more stable and consistent training.

Appendix B

Sbatch scripts

The development of SpectroBERT was made possible by the computational resources provided by HPC@POLITO (`http://www.hpc.polito.it`), which offers a SLURM environment where jobs are queued to the cluster by means of sbatch scripts. For the sake of completeness, we report some example sbatch scripts, in the hope they might be useful to those interested in replicating our experiments.

Pretraining

```
#!/bin/bash
#SBATCH --time=96:00:00
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=4
#SBATCH --partition=cuda
#SBATCH --mem=12GB
#SBATCH --gres=gpu:1

############################

# pretraining
# --dataset wavcaps or librispeech or common_voice
DATASET=wavcaps
python train.py --initialize_weights --cuda --task pre --dataset ${
    DATASET} --experiment_name ${DATASET}_pretraining
```

AQA

```
#!/bin/bash
#SBATCH --time=12:00:00
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=2
#SBATCH --partition=cuda
#SBATCH --mem=6GB
#SBATCH --gres=gpu:1

############################
# --task either 'aqa_binary' or 'aqa_open'
python train.py --initialize_weights --cuda --task aqa_binary --
    dataset clothoaqa --experiment_name clothoaqa_binary
```

SER

```
#!/bin/bash
#SBATCH --time=2:00:00
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=2
#SBATCH --partition=cuda
#SBATCH --mem=6GB
#SBATCH --gres=gpu:1

############################

# ser
# --dataset ravdess or iemocap
DATASET=ravdess
python train.py --initialize_weights --cuda --task ser --dataset ${
    DATASET} --experiment_name ${DATASET}_ser
```

Appendix C

Datasets download scripts

For the same reasons we included the sbatch scripts, we also report the bash scripts used to download the publicly-available datasets.

Note: LibriSpeech [24] and CommonVoice [25] are not included here because, although public, they are available in Hugging Face's Datasets [39] library, which makes their usage easier.

WavCaps

```
#!/bin/bash

print_usage() {
  printf "Usage: ./download_wavcaps.sh [-a] [-b] [-f] [-s]\n
  \t-a: Download AudioSet
  \t-b: Download BBC Sound Effects
  \t-f: Download FreeSound
  \t-s: Download SoundBible\n
Flags can be freely combined, for example:
  \t-abfs: Download all of the above\n"
}

printf "
==========================================
=== Downloading WavCaps dataset ===
==========================================\n"

printf "\n>>> Cloning the repository from HuggingFace <<<\n"
git lfs install
GIT_LFS_SKIP_SMUDGE=1 git clone https://huggingface.co/datasets/cvssp
    /WavCaps

```

```
printf "\n>>> Changing current working directory to the repository
    <<<\n"
cd WavCaps
pwd

printf "\n>>> Downloading all the metadata <<<\n"
git lfs pull --include "*.json"

printf "\n>>> Downloading the requested sub-datasets only <<<\n"
while getopts 'abfs' flag; do
  case "${flag}" in
    a) echo "    AudioSet_SL"
    git lfs pull --include "*/AudioSet_SL/*" ;;
    b) echo "    BBC_Sound_Effects"
    git lfs pull --include "*/BBC_Sound_Effects/*" ;;
    f) echo "    FreeSound"
    git lfs pull --include "*/FreeSound/*" ;;
    s) echo "    SoundBible"
    git lfs pull --include "*/SoundBible/*";;
    *) print_usage
       exit 1 ;;
  esac
done

printf "\n>>> Done <<<"
```

Clotho-AQA

```
#!/bin/bash

printf ">>> Downloading ClothoAQA dataset <<<\n"
mkdir ClothoAQA
cd ClothoAQA
pwd

printf "\n>>> Downloading the audio files <<<\n"
wget -nc https://zenodo.org/record/6473207/files/audio_files.zip?
    download=1 -O audio_files.zip

printf "\n>>> Downloading the metadata files <<<\n"
wget -nc https://zenodo.org/record/6473207/files/clotho_aqa_metadata.
    csv?download=1 -O clotho_aqa_metadata.csv
wget -nc https://zenodo.org/record/6473207/files/clotho_aqa_test.csv?
    download=1 -O clotho_aqa_test.csv
wget -nc https://zenodo.org/record/6473207/files/clotho_aqa_train.csv
    ?download=1 -O clotho_aqa_train.csv
```

```
wget -nc https://zenodo.org/record/6473207/files/clotho_aqa_val.csv?download=1 -O clotho_aqa_val.csv

printf "\n>>> Unzipping the audio files <<<\n"
unzip -n audio_files.zip

printf "\n>>> Fixing a typo in a filename <<<\n"
mv audio_files/souffle_me.tallique.wav audio_files/souffle_me_tallique.wav
```

RAVDESS

```
#!/bin/bash

printf ">>> Downloading RAVDESS dataset <<<\n"
mkdir RAVDESS
cd RAVDESS
pwd

printf "\n>>> Downloading the audio files <<<\n"
wget -nc https://zenodo.org/records/1188976/files/Audio_Speech_Actors_01-24.zip?download=1 -O Audio_Speech_Actors_01-24.zip

printf "\n>>> Unzipping the audio files <<<\n"
unzip -n Audio_Speech_Actors_01-24.zip
```

Appendix D

Environment details

The project was developed using Python [40] version 3.10 with the following requirements:

```
torch>=2.0
torchaudio
torchvision
pytorch-lightning>=2.0
librosa
pandas
numpy
matplotlib
transformers
datasets
tokenizers
isort
black
rich
comet_ml
pytest
torchmetrics
```