UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Dept. of Computer Systems and Computation

VISMAID: Visual Impairment Support through Multimodal AI-driven Description

Master's Thesis

Master's Degree in Artificial Intelligence, Pattern Recognition and Digital Imaging

AUTHOR: Camas Nájera, Ramsés

Tutor: Casacuberta Nolla, Francisco

ACADEMIC YEAR: 2023/2024

Universitat Politècnica de València
Dept. of Computer Systems and Computation
Master's Degree in Artificial Intelligence, Pattern Recognition and Digital Imaging

VISMAID: Visual Impairment Support through Multimodal AI-driven Description

Master's Thesis

Ramsés Alejandro Camas Nájera

Supervised by:
Dr. Casacuberta Nolla, Francisco

Academic Course 2023/2024

Acknowledgements

This work is not mine alone; it has been made possible by the support of many people who helped make this dream a reality. First and foremost, I want to express my deepest gratitude to my parents, Raúl Camas and Sugey Nájera, for supporting me from the moment I wrote my first line of code and always encouraging me to try new things, even when I was afraid. I also want to acknowledge my grandfather on my mother's side, who was the inspiration behind this work. As one of the visually impaired individuals whom this project aims to help, he motivated me to persevere. This year, I faced the fear of almost losing him to a stroke, while being nearly 10,000 km away. This project is my way of supporting him despite the distance between us.

I could not have accomplished this without my girlfriend, Ada Chanona, who has supported me in every project and adventure since 2020. Without her, it would have been nearly impossible to cope with the loneliness of being in a new country for the first time. Thanks to her, I have developed a more empathetic side that helped make this work human-centered. I will always be grateful for everything she has done for me and for helping me become a better person.

I am also deeply grateful to my aunt, Marisol Camas, who was instrumental in enabling me to pursue this master's degree abroad; without her, I might still be in Mexico. I extend my thanks to Dr. Francisco Casacuberta for his patience and support throughout this project. Although our contact was not constant, he always did his best to help me whenever needed. I would like to thank the Valencian Graduate School and Research Network of Artificial Intelligence (valgrAI) for partially funding this project and supporting me in completing my master's degree.

I extend my heartfelt thanks to my dearest friends, who were devastated on the day I left Mexico to pursue this degree: to Luis Hernández, for sharing the same broken sense of humor, for being the best roommate during my last year of college, and for standing by my side at every hackathon and programming competition; to Viviana Rojo, for always sharing a laugh, even on the worst days, for always having something interesting to discuss, and for never doubting me; and to Victor Rios, for helping me through all the math subjects in college, without whom graduation would have been much harder, and for all the times I kept him waiting.

Lastly, I want to acknowledge my younger brother, Kaleb Camas, who is just beginning his college journey, to show him that it is possible to achieve your dreams, no matter how far from home you may be. I also wish to thank my brothers by choice, José Nango and Iván Rincón, for always being there, for decades of friendship, and for continually encouraging me to improve and become better each day. This is just one more step forward.

Thanks to all these people, I am where I am today. As Isaac Newton said, "If I have seen further, it is by standing upon the shoulders of giants." This work is dedicated to all of you.

Abstract

This thesis presents ”VISMAID” (Visual Impairment Support through Multimodal AI-driven Description), a novel solution designed to assist individuals with visual impairments. Leveraging state-of-the-art multimodal AI models, VISMAID integrates audio, vision, and language processing capabilities to enable a user-friendly experience on mobile devices. The system architecture employs a combination of Automatic Speech Recognition (ASR), Vision-Language Models (VLMs), and Text-to-Speech (TTS) technologies, allowing users to interact with their surroundings through voice commands and receive audio descriptions of visual content captured by their device's camera.

The proposed solution addresses several technical challenges, such as optimizing model performance for limited hardware, reducing latency, and managing energy consumption. The implementation utilizes advanced techniques like Low-Rank Adaptation (LoRA) and quantization methods to enhance model efficiency and adaptability on mobile platforms. Evaluation results demonstrate that VISMAID achieves a balance between computational efficiency, accessibility, and real-time performance, supporting multiple use cases, including Visual Question Answering (VQA) and Image Captioning.

Furthermore, the research emphasizes the importance of user-centered design, ensuring the solution is intuitive and practical for real-world applications. Extensive user testing and feedback loops were incorporated throughout the development process to refine the functionality and usability of the system. The outcomes highlight the potential for VISMAID to serve as a powerful tool in assisting visually impaired individuals, paving the way for further innovations in AI-driven accessibility solutions.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

As humans, we perceive our environment through our senses, we see, smell, hear, touch and taste the thing around us to understand the world we live in. Humanity has adapted their surroundings for the human senses, from the use of artificial lighting to overcome the darkness to alarm systems on homes or even car horns. However, there are some cases where a human being lost one of its senses, or start losing one of them periodically, for those cases there are traditional alternatives, such as the Braille Alphabet, a tactile writing system used by people who are visually impaired [41].
This work focus on being a support for people who have lost or is losing their sense of vision through state of the art technology like vision language models.

Chapter 2

Theoretical Framework

In this chapter, we introduce the concept of multimodality, Multimodal Machine Learning and multimodal tasks. Some of the state of the art models known as Vision Language Models, the complex task known as Visual Question Answering and how was defined, a surface analysis on Text ot Speech and Speech to Text models, including state of the art models such as Whisper [36], how this massive models can be adapted to work on limited hardware by using quantization and how to extend the domain of a model through the use of multiple experts.

2.1 Basics

Machine learning is a branch of computer science that concentrates on creating algorithms based on sets of examples of a particular phenomenon. These examples may originate from natural occurrences, be manually created by humans, or be produced by another algorithm. [8]

As is defined by Tom Mitchell on his book *"Machine Learning"*[28]:

*A computer program is said to learn from experience **E** with respect to some class of task **T**, and performance measure **P**, if its performance at task in**T**, as measured by **P**, improves with experience **E**.*

This means there are many kinds of machine learning systems, depending on the nature of the tasks **T**, the nature of the performance measure **P** and the nature of the experience **E** also known as training signal. [29]

2.1.1 Supervised Machine Learning

As described by Kevin P. Murphy in his book *"Probabilistic Machine Learning: An Introduction"*. [29]

The most common form of machine learning is supervised learning. In this method, the objective or task T is to learn a function f that maps inputs $x \in X$ to outputs $y \in Y$. The inputs, also known as features, covariates, or predictors, are generally represented as a vector of numbers with a fixed number of dimensions. These inputs could be measurements like a person's height and weight or the pixel values in an image. Here, $X = \mathbb{R}^D$, where D represents the number of dimensions of the vector (i.e., the number of input features). The output y is also referred to as the *label, target, or response*. The learning experience E consists of a set of N input-output pairs, $D = \{(x_n, y_n)\}_{n=1}^{N}$, which is called the *training set* (where N is the sample size). The performance measure $P$is determined by the type of output being predicted.

For classification problems, the output is a set of C unordered and mutually exclusive label known as classes, while for regression task the output can be a real number and even for other tasks the output can be a more complex structure, such as a vector, a matrix, a tree or a graph. [8]

2.1.2 Neural Networks

Dense neural networks, also referred to as feedforward neural networks or multilayer perceptrons (MLPs), are foundational models in the field of deep learning. The goal of a feedforward network is to approximate a target function f^*. For example, in a classification problem, $y = f^*(x)$ denotes a function that maps an input x to its corresponding class y. A feedforward network defines a function $y = f(x; \theta)$ and aims to learn the best parameter values θ to closely approximate this function. These networks are called "feedforward" because the information flows in a single direction—from the input x, through the intermediate layers representing the computation of f, and ultimately to the output y. Unlike other models, they lack feedback connections, meaning the outputs are not looped back into the network [16].

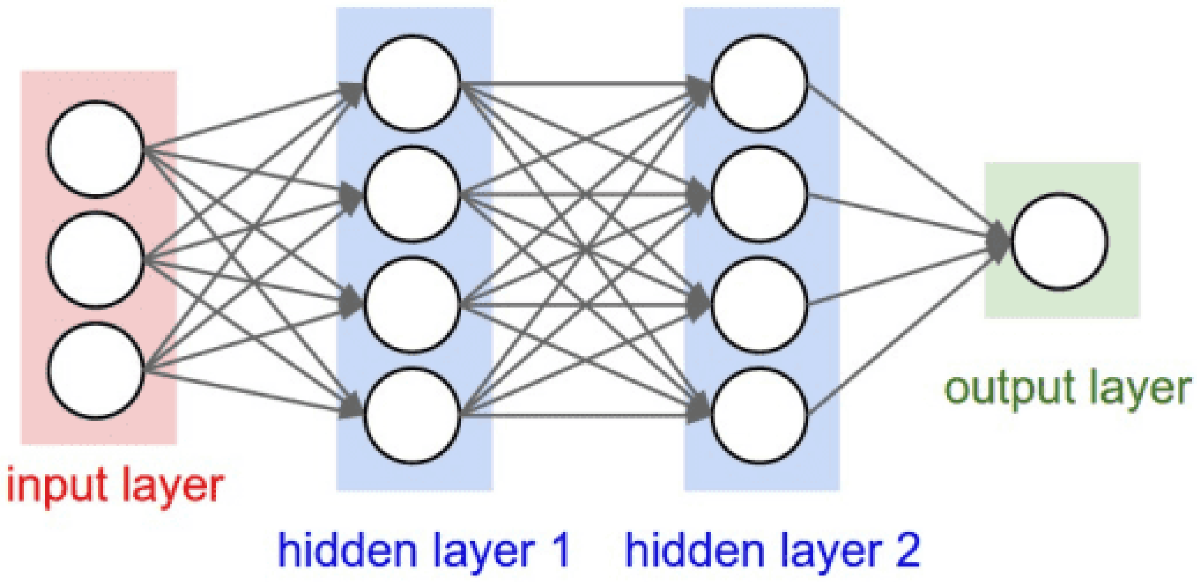

Figure 2.1: Example of the structure of a Feed Forward Neural Network.

Feedforward networks are extremely important for machine learning practitioners and are the foundation of many significant commercial applications. For example, convolutional networks, which are specialized types of feedforward networks, are used for tasks like recognizing objects in images. Additionally, feedforward networks serve as a conceptual foundation for recurrent networks, which are crucial in many natural language processing applications. The term "network" is used because feedforward neural networks are generally represented by combining many different functions. The model is associated with a directed acyclic graph that illustrates how these functions are composed.

For instance, we might have three functions $f^{(1)}$, $f^{(2)}$, and $f^{(3)}$ connected in a sequence to form $f(x) = f^{(3)}(f^{(2)}(f^{(1)}(x)))$. These chain-like structures are the most common architecture used in neural networks. In this context, $f^{(1)}$ is known as the *first layer* of the network, $f^{(2)}$ is the *second layer*, and so forth. The total number of layers in the chain determines the *depth* of the model, which is where the term "deep learning" originates. The last layer of a feedforward network is referred to as the *output layer*. During training, the objective is to make $f(x)$ closely approximate $f^*(x)$.

The training data offers noisy, approximate samples of $f^*(x)$ evaluated at different points. Each input x is paired with a label $y \approx f^*(x)$. These training samples directly specify what the output layer should produce for each input x; it should yield a value that closely matches y. However, the training data does not provide explicit instructions for the behavior of the intermediate layers. The learning algorithm must figure out how to configure these hidden layers to produce the correct output. Since the desired output for each hidden layer is not specified by the training data, these layers are referred to as *hidden layers*.

These networks are called *"neural"* due to their loose inspiration from neuroscience. Each hidden layer within the network is generally represented by a vector, with the number of dimensions of these hidden layers defining the *width* of the model. Each element of this vector can be thought of as analogous to a neuron. Rather than considering the layer as a single function mapping one vector to another, it can also be viewed as a collection of several *units* operating in parallel, where each unit represents a function that maps a vector to a scalar. Each unit behaves like a neuron by receiving inputs from multiple other units and computing its own activation level. The concept of multiple layers of vector-based representations is motivated by neuroscience. Similarly, the choice of functions $f^{(i)}(x)$ for computing these representations is loosely inspired by the functional roles performed by biological neurons.

2.1.3 Convolutional Neural Networks

Convolutional neural networks (CNNs) are a specialized type of neural network architecture designed to process data arranged in a grid-like structure [23]. Examples of such data include time-series data, which can be viewed as a one-dimensional grid where samples are collected at uniform time intervals, and image data, which can be considered a two-dimensional array of pixels. CNNs have achieved significant success across various practical domains. The term "convolutional neural network" originates from the use of an operation called convolution, which is a specific kind of linear transformation. Fundamentally, CNNs are neural networks that apply convolution operations in place of general matrix multiplication in at least one of their layers.

To recognize specific patterns in CNNs, a small regression model learns the parameters of a matrix $\mathbf{F}$ with dimensions $p \times p$, where p represents the size of a local patch. Each layer in a CNN is made up of multiple convolutional filters, each with its own bias term, much like a layer in a standard feedforward neural network (FFNN) is composed of numerous neurons. Each filter in the first (leftmost) layer slides, or convolves, across the input image from left to right and top to bottom, calculating the convolution at each location.

The parameters within the filter matrix $\mathbf{F}$ for every filter in each layer, along with the bias b, are learned through gradient descent using backpropagation to minimize the loss function based on the training data. A nonlinear activation function is then applied to the sum of the convolution output and the bias term. The ReLU function is commonly used in hidden layers, while the activation function for the output layer is determined by the specific task requirements. Given that each layer l can contain multiple filters of size

l, the output from convolutional layer l comprises l matrices, one for each filter.

When a CNN contains consecutive convolutional layers, the following layer $l + 1$ interprets the output of the prior layer l as a set of l-dimensional image matrices, known as a **volume**. Each filter in layer $l+1$ then performs convolutions over this entire volume. The convolution of a patch within a volume is simply the sum of the convolutions of the matching patches from each separate matrix within the volume.

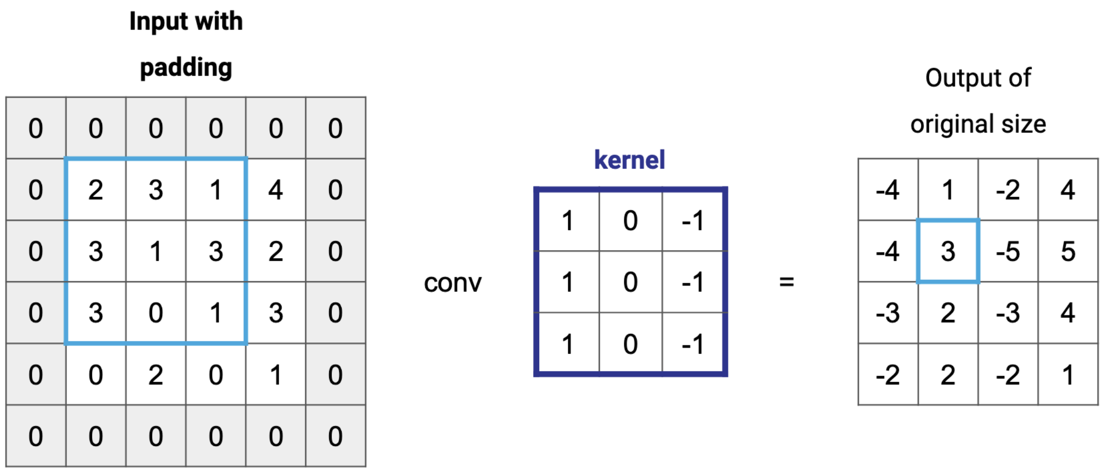

Figure 2.2: Examples of a convolution with padding.

Two other key properties of convolution are **stride** and **padding**. The stride is the step size of the sliding window that moves across the input. Padding is a technique used to produce a larger output matrix by adding extra cells around the edges of the input volume before applying the convolution. These additional cells, which typically contain zeros, increase the size of the output. Alongside these properties, **pooling** is used as a technique that, instead of applying a learnable filter to an input matrix or volume, uses a fixed operator. The pooling layer takes the output from a convolution as its input and produces an output volume with the same depth as the input. [8]

Convolutional networks have played a significant role in the development of deep learning. They are a prime example of successfully applying insights from studying the brain to machine learning tasks. Convolutional networks were among the first deep learning models to achieve good performance, even before arbitrary deep models were considered feasible. They were also among the first neural networks to be used for solving important commercial applications and continue to be at the forefront of commercial uses of deep learning today.

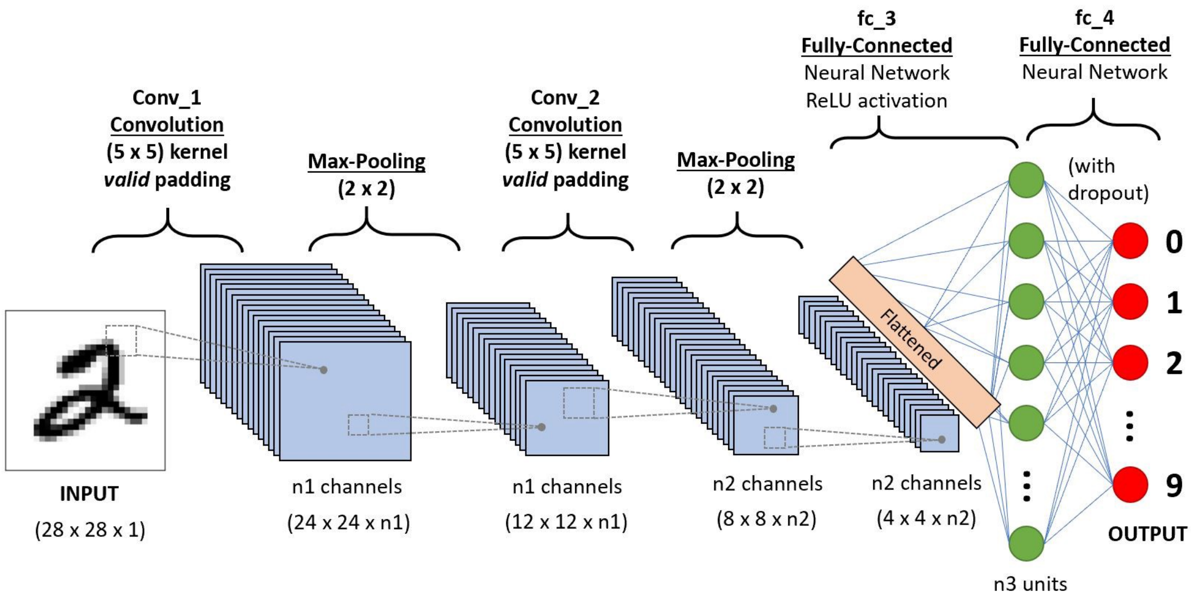

Figure 2.3: Visual Representation of a Convolutional Neural Networks

Convolutional networks were also among the earliest deep networks trained using backpropagation. The reason for their success, while general backpropagation networks were thought to have failed, is not entirely clear. One potential reason is that convolutional networks are more computationally efficient than fully connected networks, allowing for more extensive experimentation and refinement of their implementations and hyperparameters. Larger networks also appear to be simpler to train. With today's hardware, large fully connected networks achieve satisfactory performance on numerous tasks, even when using datasets and activation functions that were common during the era when fully connected networks were believed to be less effective. [16]

2.1.4 Recurrent Neural Networks

A recurrent neural network (RNN) maps input sequences to output sequences in a state-dependent manner, where the output y_t depends on both the input x_t and the hidden state h_t, which updates as the sequence is processed. RNNs are suitable for tasks like sequence generation, classification, and translation [29].

While convolutional neural networks (CNNs) are designed for grid-like data, such as images, RNNs are optimized for sequences of values $x^{(1)}, \ldots, x^{(\tau)}$. RNNs can handle much longer sequences than models not designed for sequential data and can also process sequences of variable lengths.

Parameter sharing allows RNNs to generalize across sequences of different lengths, avoiding the need for separate parameters at each time step, which would limit generalization to new sequence lengths and hinder learning from different positions in time. In RNNs, each output depends on previous outputs, using the same update rule across all steps, resulting in a deep computational graph [16].

RNNs can map input sequences to output sequences of varying lengths, useful in applications like speech recognition, machine translation, and question answering, where the input and output sequences often differ in length. The input to the RNN is referred to as the "context," represented as a vector or vectors summarizing the input sequence $X = (x^{(1)}, \dots, x^{(n_x)})$.

The encoder-decoder, or sequence-to-sequence, is a basic RNN architecture that handles variable-length input and output sequences. It consists of: (1) an encoder RNN that processes the input and generates the context C, usually from its final hidden state, and (2) a decoder RNN that, based on this context, produces the output sequence $Y = (y^{(1)}, \dots, y^{(n_y)})$. This model accommodates varying lengths for n_x and n_y, unlike earlier architectures requiring $n_x = n_y = \tau$. Both RNNs are jointly trained to maximize the log probability $\log P(y^{(1)}, \dots, y^{(n_y)} | x^{(1)}, \dots, x^{(n_x)})$ over all training pairs, using the encoder's final state h_{n_x} as the context for the decoder.

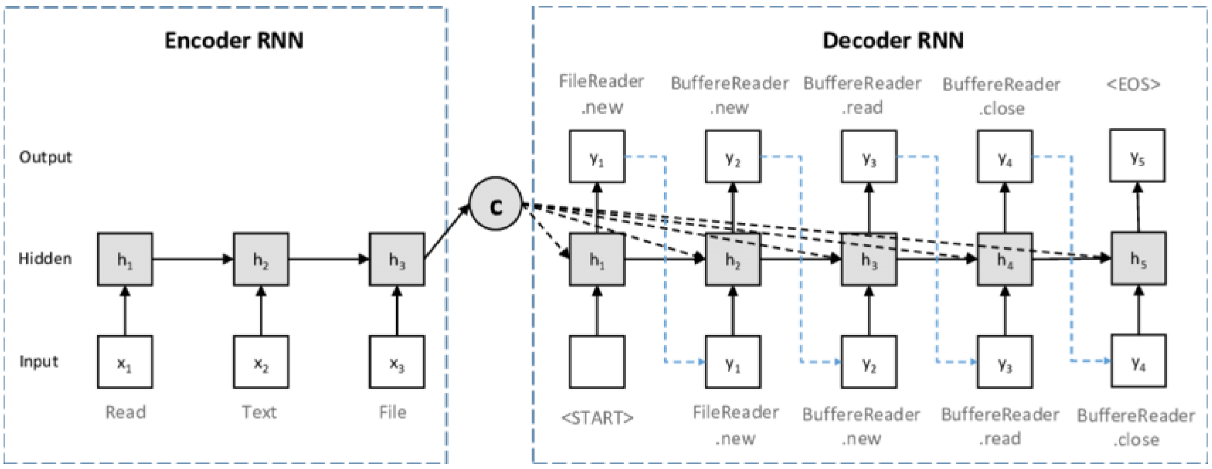

Figure 2.4: Example of an Encoder-Decoder Architecture

The input can be supplied either as the initial state of the decoder RNN or connected to its hidden units at each time step. There is no requirement for the encoder and decoder to have hidden layers of the same size. However, a clear limitation of this architecture arises when the context C, output by the encoder RNN, has a dimension that is too small to adequately summarize a long sequence. This issue was observed by Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio [5] in the context of machine translation. They proposed making C a variable-length sequence rather than a fixed-size vector. Additionally, they introduced an attention mechanism that learns to associate elements of the sequence C with elements of the output sequence.

2.1.5 Transformers

Attention Mechanism

Capturing all the semantic details of a very long sentence, such as one with 60 words, using a fixed-size representation is quite challenging. This can be done by training a sufficiently large RNN for a long enough period, but a more efficient approach is to first read the entire sentence or paragraph to understand the context and main idea, and then generate the translated words one at a time. In this process, the model focuses on different parts of the input sentence to extract the necessary semantic details for producing each subsequent output word. The attention mechanism is used to focus on specific parts of the input sequence at each step.

An attention-based system can be thought of as consisting of three main components:

- A process that reads the raw input data (such as words in a source sentence) and converts them into distributed representations, with each word position associated with a **feature vector**.
- A list of feature vectors that store the output of this reader, which can be seen as a **memory** containing a sequence of facts. This memory allows retrieval of information in any order, without needing to access all elements.
- A process that uses the content of this memory to perform a task sequentially, with the ability to pay **attention** on the content of one or several memory elements at each time step, each with different weights. This component is responsible for generating the translated sentence.

When words in a sentence written in one language are aligned with their corresponding words in a translated sentence in another language, it becomes possible to relate their corresponding word embeddings. [16]

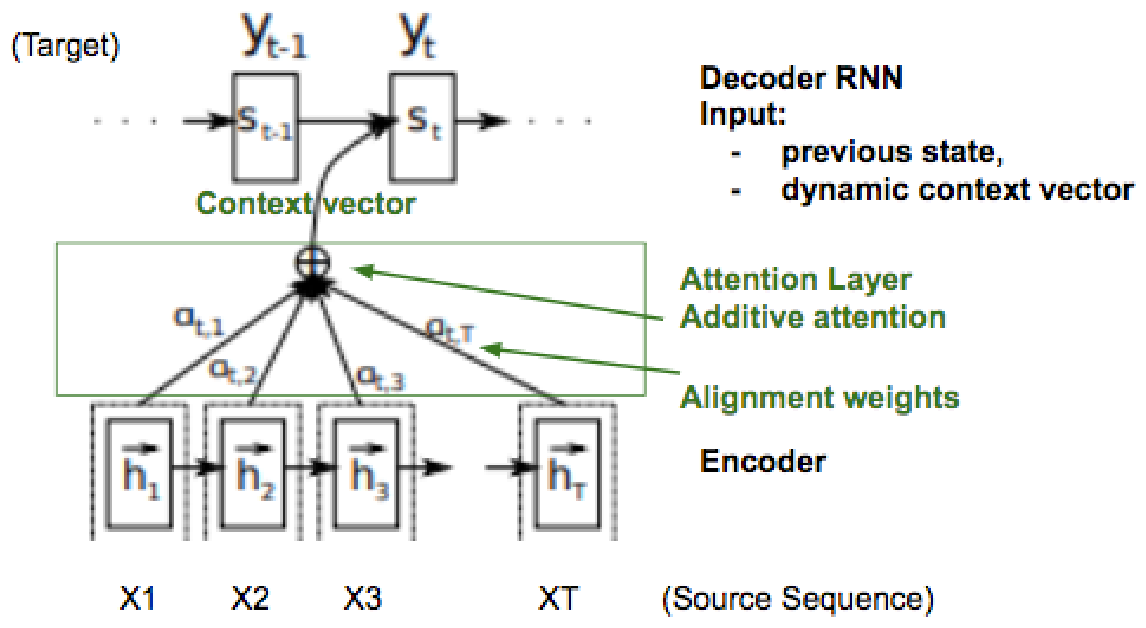

Figure 2.5: Attention Mechanism proposed by D. Bahdanau [5]

The Transformer model, as introduced in "Attention Is All You Need"[40], relies entirely on the attention mechanism to compute representations of its input and output without using recurrent or convolutional networks. The attention mechanism allows the model to focus on different parts of the input sequence when predicting a particular output, making it possible to capture long-range dependencies more effectively.

The Transformer model uses the attention mechanism to compute representations of its input and output sequences, discarding the need for recurrent or convolutional networks. The key components of the attention mechanism used in the Transformer are self-attention and multi-head attention.

Self-Attention: Self-attention, also known as intra-attention, computes a representation of a sequence by relating different positions of the same sequence. Given a set of queries Q, keys K, and values V, all of dimension d_k, the self-attention function is defined as:

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

Each query vector is compared against all key vectors to calculate attention scores, which are scaled by the square root of the dimensionality d_k to avoid large gradients. The softmax function normalizes these scores, which are used to compute a weighted sum of the value vectors, producing the attention output.

Scaled Dot-Product Attention: The scaled dot-product attention involves:

- Calculating the dot products between the query and key vectors.
- Scaling the dot products by $\frac{1}{\sqrt{d_k}}$.
- Applying the softmax function to obtain normalized weights.
- Multiplying these weights by the value vectors to produce the final attention output.

Multi-Head Attention: Multi-head attention enables the model to simultaneously focus on information from different subspaces of representations. Rather than using a single attention function, the Transformer linearly maps the queries, keys, and values h times using distinct learned projections and then applies the attention mechanism in parallel. The results from each of these heads are concatenated and subsequently passed through a linear transformation to produce the final output:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O$$

where each attention head is calculated as:

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

In this context, W_i^Q, W_i^K, W_i^V, and W^O represent the learnable projection matrices.

Transformer Architecture Overview

The Transformer model consists of an encoder-decoder structure, similar to many neural sequence transduction models. However, it departs from traditional designs by discarding recurrence and convolution entirely, relying solely on attention mechanisms and position-wise feed-forward networks.

- **Encoder-Decoder Structure:**
  - **Encoder:** The encoder is composed of a stack of $N = 6$ identical layers. Each layer has two main sub-layers:
    - A multi-head self-attention mechanism.
    - A position-wise fully connected feed-forward network.

    Each sub-layer is surrounded by a residual connection followed by layer normalization.

  - **Decoder:** The decoder is also composed of a stack of $N = 6$ identical layers, but with an additional sub-layer between the two main sub-layers. This third sub-layer performs multi-head attention over the output of the encoder stack, allowing the decoder to focus on relevant parts of the input sequence. The self-attention sub-layer in the decoder is masked to ensure that the prediction for each position depends only on the known outputs at preceding positions.

- **Attention Mechanisms:**
  - **Self-Attention:** Allows the model to attend to all positions in the input sequence to capture dependencies, regardless of their distance.
  - **Multi-Head Attention:** Improves the model's ability to capture different types of relationships by projecting queries, keys, and values into multiple subspaces and performing attention in parallel. This allows the model to attend to different aspects of the input simultaneously.

- **Positional Encoding:**
  - Since the Transformer lacks any recurrence or convolution to handle sequential data, it uses positional encodings to inject information about the relative or absolute position of tokens in the sequence. These encodings are added to the input embeddings and are based on sinusoidal functions, which help the model learn positional relationships.

- **Position-Wise Feed-Forward Networks:**
  - Each layer in both the encoder and decoder contains a fully connected feed-forward network that is applied independently to each position. This consists of two linear transformations with a ReLU activation in between. This structure enables the model to transform attention outputs and introduces non-linearity.

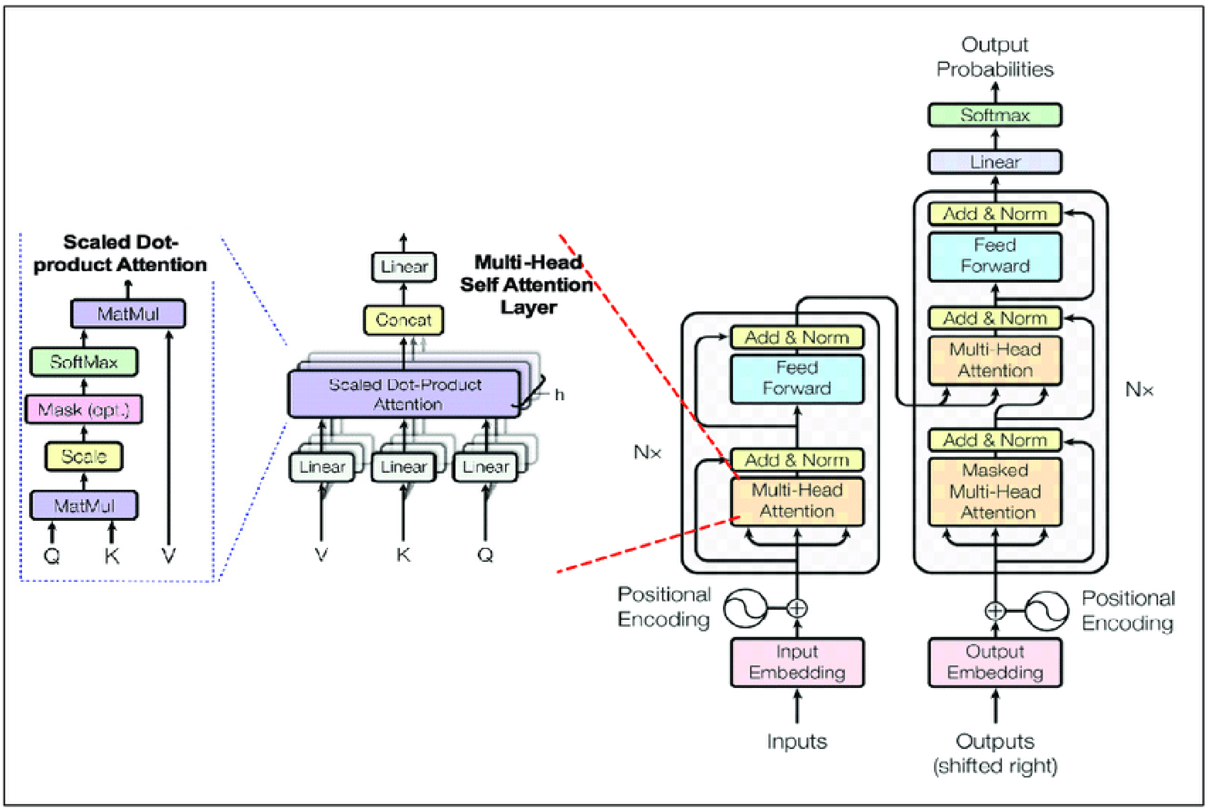

Figure 2.6: Visual Representation of the Transformer Architecture

Improvements Over Previous Models

The Transformer architecture brings several key improvements compared to previous neural network models, particularly RNN-based models like LSTM (Long Short-Term Memory) [19] and GRU (Gated Recurrent Unit) [9], as well as CNN-based models like ConvS2S [15]:

- **Parallelization:** One of the major limitations of RNNs is their sequential nature, which makes it difficult to parallelize training across long sequences. Each step in an RNN depends on the completion of the previous step, which slows down training. The Transformer, by contrast, allows for complete parallelization within a sequence, as all the tokens are processed simultaneously through the self-attention mechanism. This leads to significant speed-ups in training, allowing the Transformer to be trained more efficiently on modern hardware like GPUs.

- **Long-Range Dependencies:** RNNs struggle with long-range dependencies due to the vanishing gradient problem, where information from earlier time steps is lost as it moves through many layers. Although LSTM and GRU models mitigate this issue, they still face limitations with very long sequences. The Transformer's self-attention mechanism, however, allows it to directly attend to any position in the input sequence, regardless of its distance from the target position. This enables the model to capture long-range dependencies more effectively and improves the quality of representations for tasks such as translation and text generation.

- **Computational Efficiency:** CNN-based models like ConvS2S reduce sequential computation, but they still suffer from the increased complexity of modeling long-range dependencies. In contrast, the Transformer's attention mechanism has a constant path length between input and output positions, making it more efficient in handling long dependencies. While the computational complexity of self-attention is $O(n^2)$, it is still more efficient than the $O(n)$ complexity of recurrent layers when the sequence length n is smaller than the representation dimension d, which is often the case in many practical scenarios.

- **Better Performance on Benchmarks:** The Transformer has achieved state-of-the-art results on several benchmark tasks. For example, in machine translation tasks, the Transformer model significantly outperforms RNN-based models, including Google's Neural Machine Translation (GNMT) system, and convolutional models like ByteNet and ConvS2S, both in terms of quality (measured by BLEU scores) and computational cost. For instance, the Transformer achieved a BLEU score of 28.4 on the WMT 2014 English-to-German translation task, outperforming all previously reported models, including ensembles, while requiring less training time.
- **Flexibility and Generalization:** The Transformer architecture generalizes well to other tasks beyond machine translation. It has shown competitive performance in tasks like constituency parsing and can be adapted to modalities other than text, such as image and audio processing. Its ability to handle variable-length sequences, capture complex dependencies, and process information in parallel makes it a versatile model for a wide range of applications.

2.2 Multimodality

As said before, humans experience the world multiple modalities, when we speak about modality refers to the manner in which something occurs or is perceived. A problem is considered multimodal when it involves data in multiple different modalities. For Artificial Intelligence to advance in comprehending the world, it must be capable of interpreting these diverse multimodal signals collectively.[6]

We talk about Multimodal Machine Learning where the information receiver and processor is a Machine Learning system, this field seeks to develop models that can handle and connect information from various modalities. This is a dynamic and rapidly growing field that spans multiple disciplines and holds immense potential. This field presents unique challenges for computational researchers due to the diverse nature of the data involved. Learning from multiple modalities allows for the identification of connections between different types of data and enables a deeper understanding of natural phenomena.

But this definition of multimodality can be discussed, some literature define modality being task related instead of information related.[32] In the Multimodal Machine Learning field, it is widely recognized that vision constitutes one modality and language represents another, usually presented in the form of text. [21] However, humans can hear the speech, which the information that it transfers is language. In the same sense, language can be transmitted through touch with the use of Braille or vision by reading something. This means language can be represented using many modalities of data, with this definition the modalities are human-centered, however Letitia Parcalabescu (Parcalabescu et al., 2021) [32] suggests that the definition of multimodality in Multimodal Machine Learning should be tied to the specific task the Machine Learning system is designed to solve. The task itself dictates which information is relevant and how that information can be most efficiently represented. In contrast, the human-centered definitions attempt to define multimodality without considering specific tasks, instead focusing on categories such as human experience, media types, forms of representation, and data encodings. So, we use the following definition for multimodality:

A machine learning ***task*** *is multimodal when inputs or outputs are* ***represented*** *differently or are composed of distinct types of atomic units of information* [32].

2.3 Multimodal Deep Learning

As its name suggests, Multimodal Deep Learning focus on the fusion and analysis of data from multiple modalities, such as text, images, video, audio, and sensor data. Thus, in order to create a more complete and much complex representation of the data, all of this using multiple deep neural networks, each designed to specialize in processing a specific modality and then merge the output using different fusion techiniques to form a combined representation of the data[33]

The main objective of Multimodal Deep Learning is to develop a shared representation space that effectively captures complementary information from various modalities. This unified representation can be utilized for a range of tasks, including image captioning, speech recognition, and vision question answering.

Some fusion techiniques in Multimodal Deep Learning are: early fusion, late fusion, or hybrid fusion. Early fusion involves combining raw data from different modalities into a single input vector and feeding it directly into the network. Late fusion, in contrast, trains separate networks for each modality and integrates their outputs at a later stage. Hybrid fusion incorporates aspects of both early and late fusion, creating a model that is more adaptable and flexible.

Multimodal deep learning models generally consist of several unimodal neural networks, each designed to handle a specific input modality independently. For example, an audiovisual model might include one network for processing audio data and another for visual data. This separate handling of each type of input is referred to as encoding.

After each modality has been encoded separately, the information extracted from each one needs to be combined or fused. This fusion can be achieved using various techniques, ranging from simple concatenation to more sophisticated attention mechanisms. Effective fusion of multimodal data is crucial for the success of these models. Finally, a "decision" network receives the fused information and is trained to perform the specific task.

Typically, multimodal architectures consist of three main components:

- **Unimodal encoders** that process and encode each individual modality, usually one encoder for each type of input.
- **A fusion network** that integrates the features obtained from each modality during the encoding phase.
- **A classifier** that takes the fused information and makes predictions based on it.

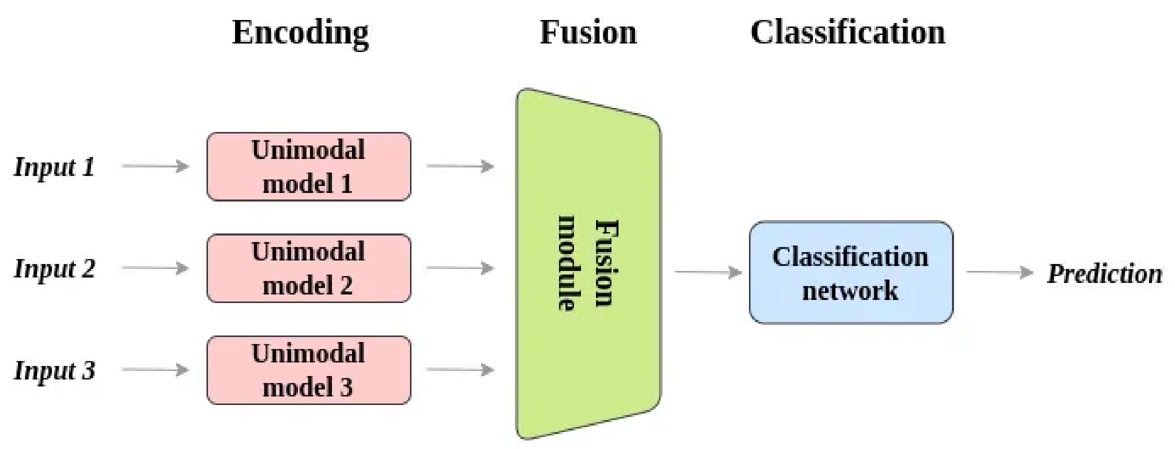

Figure 2.7: Illustration of a base multimodal architecture.

2.3.1 Encoding Sate

The encoder extracts features from input data across different modalities, such as images, audio, and text, and transforms them into a shared representation for further processing. Each modality has its own encoder that converts input data into feature vectors, which are then combined into a unified representation. This can be done through concatenation or attention mechanisms to emphasize the most relevant information. The encoder's main goal is to capture the underlying patterns and relationships within the multimodal data, enabling the model to make accurate predictions or generate outputs.

2.3.2 Fusion Stage

The fusion module integrates information from different modalities, such as text, images, and audio, into a single representation suitable for tasks like classification, regression, or generation. The structure of the fusion module can vary based on the architecture and task requirements.

One method involves using a weighted sum of the features from each modality, with the weights learned during training. Another method is to concatenate the features and use a neural network to form a combined representation. Attention mechanisms can also be applied to determine which modality should be prioritized at each step.

The primary objective of the fusion module is to combine complementary information from multiple modalities to create a stronger and more informative representation, which enhances performance in tasks such as video analysis, where both visual and audio data are important.

2.3.3 Classification

The classification module uses the combined representation created by the fusion module to make a prediction or decision. Its design and approach can vary based on the specific task and data being processed.

Typically, the classification module is a neural network that processes the joint representation through one or more fully connected layers to arrive at the final prediction. These layers may include non-linear activation functions, dropout, and other techniques to enhance generalization and prevent overfitting.

The output of the classification module is task-specific. For example, in multimodal sentiment analysis, it might produce a binary output indicating whether the input is positive or negative. In multimodal image captioning, the output could be a descriptive sentence of the image content.

The classification module is usually trained using supervised learning, where the model's parameters are optimized based on input data and their corresponding labels, often using gradient-based methods like stochastic gradient descent.

The classification module is essential in multimodal deep learning, using the fused representation to make accurate predictions or decisions.

2.4 Vision Language Models

Vision Language Models are a subset of Large Language Models which can learn simultaneously from images and texts to tackle many tasks, from visual question answering to image captioning. This familiy of models belong to the sub field of Multimodal Machine Learning known as **Multimodal Deep Learning**.

Vision-language models are generally defined as multimodal models designed to learn from both images and text. These generative models take inputs in the form of images and text and produce text-based outputs. Large vision-language models excel at generalizing across a wide range of tasks without specific training, demonstrating strong zero-shot capabilities. This means they can perform tasks they were not explicitly trained for by leveraging their understanding of both visual and textual data. [31]

These models are versatile and can handle various types of images, such as documents, web pages, and more. Their use cases include conversing about images, recognizing images based on instructions, answering questions about visual content, understanding documents, creating image captions, and other related tasks. Additionally, some vision-language models can also understand and interpret spatial relationships within an image.

There are several methods for pretraining a vision-language model, with the key being to unify the image and text representations and use them as input for a text decoder to generate outputs. Typically, the most popular models include an image encoder, an embedding projector (often a dense neural network) to align the image and text representations, and a text decoder, arranged in this sequence. Different models adopt different approaches to training.

For example, the LLaVA model uses a CLIP image encoder, a multimodal projector, and a Vicuna text decoder. The researchers utilized a dataset containing images and captions, leveraging GPT-4 to create questions associated with those captions and images. The image encoder and text decoder were kept unchanged, while only the multimodal projector was trained to align the visual and textual features. This was achieved by inputting images and the generated questions into the model and then comparing its output to the actual captions. Once the projector was pre-trained, the image encoder remained frozen, and the text decoder was unfrozen. The projector and decoder were then trained jointly. This pre-training and fine-tuning method is the most widely used strategy for developing vision-language models.

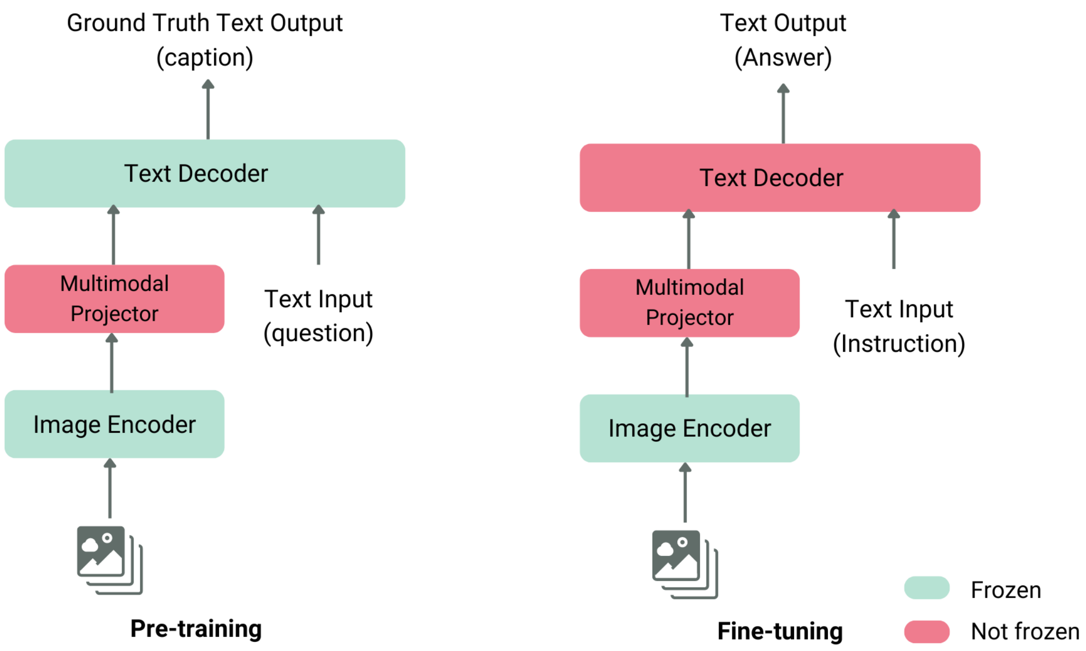

Figure 2.8: Structure of a Vision Language Model

2.5 Visual Question Answering

Visual Question Answering (VQA) is a task in the field of artificial intelligence that combines computer vision and natural language processing to answer questions about the content of an image. The VQA task requires an AI system to take as input an image and a free-form, open-ended, natural-language question related to that image and then produce an accurate, natural-language answer.

The questions in VQA are designed to be diverse and cover a wide range of topics, selectively targeting different areas of an image, including its background details and underlying context. They may require various levels of understanding and reasoning about the visual content, such as recognizing objects, identifying activities, understanding spatial relationships, and using common-sense knowledge. [3]

VQA is distinguished by its open-ended nature. Unlike tasks that generate generic image captions, VQA requires specific answers to specific questions. This specificity demands a detailed understanding of the visual content and the capability to integrate multiple types of knowledge, including fine-grained recognition (e.g., identifying the kind of cheese on a pizza), object detection (e.g., counting the number of bikes), activity recognition (e.g., determining whether a person is crying), and knowledge base reasoning (e.g., inferring whether a pizza is vegetarian).

The task is highly relevant for practical applications, such as assisting visually impaired users by providing detailed descriptions of images in response to their specific questions. VQA systems could also be useful in intelligence analysis, where specific information from visual data needs to be extracted quickly and accurately.

VQA presents a range of challenges that make it an ”AI-complete” task, meaning it encompasses multiple aspects of intelligence, including vision, language understanding, and reasoning. For instance, to answer a question like ”What is just under the tree?” a VQA system must not only recognize the objects in the image but also understand their spatial arrangement relative to the tree. Similarly, for a question such as ”Does this person have 20/20 vision?" the system must apply common-sense reasoning and potentially use external knowledge to infer an answer.

An important feature of VQA is its amenability to automatic evaluation. Many VQA answers are concise, consisting of just a few words or even a single word (e.g., "yes" or "no"), which makes it straightforward to measure accuracy by comparing the system's output to a set of ground truth answers provided by humans.

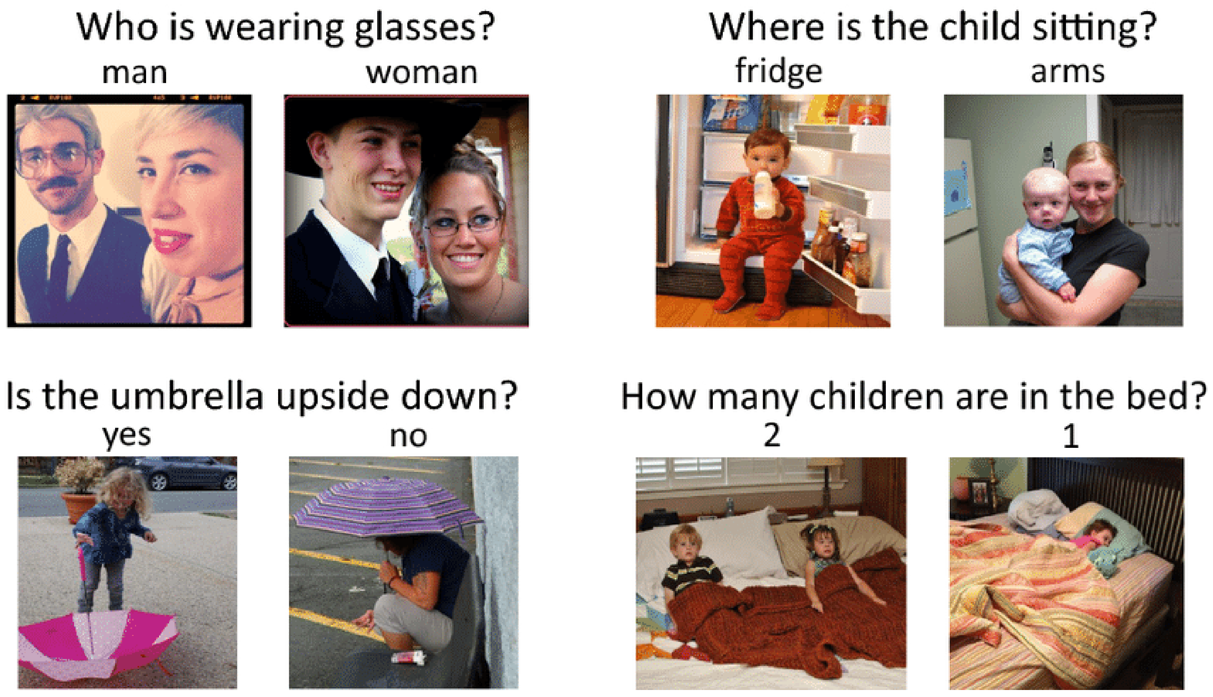
Who is wearing glasses?
man
woman
Where is the child sitting?
fridge
arms
Is the umbrella upside down?
yes
no
How many children are in the bed?
2
1

Figure 2.9: Examples from the VQA v2 dataset [17]

To support research in this area, a large dataset has been developed, which includes approximately 0.25 million images, 0.76 million questions, and 10 million answers. This dataset enables the development and benchmarking of various VQA algorithms and models. The dataset includes both open-ended and multiple-choice questions, which further facilitates the evaluation of VQA systems.

Overall, VQA serves as a compelling challenge for AI researchers, requiring advancements in visual understanding, natural language processing, and reasoning, with the ultimate goal of creating systems that can interact with and interpret visual data in a way that is comparable to human understanding.

2.6 Quantization and Optimization

Many applications in natural language processing, and increasingly in computer vision, rely on modifying a large-scale, pre-trained language model to handle various downstream tasks. This modification is generally accomplished through fine-tuning, which involves updating all the parameters of the pre-trained model. A major limitation of fine-tuning is that the modified model maintains the same parameter count as the original model [20]. Quantization comprises a collection of techniques aimed at reducing the precision of model weights, thereby decreasing the model's size and accelerating training in deep learning models. [30] Some quantization and optimization techniques are **LoRa**, 8-bit

Quantization and **QLoRa** that can quantize Transformer based models and are great for Large Language Models and Vision Language Models

2.6.1 Low-Rank Adaptation (LoRA) of Large Language Models

Low-Rank Adaptation (LoRA) is an approach aimed at efficiently adapting large pre-trained language models to different downstream tasks while minimizing computational and memory expenses. LoRA accomplishes this by keeping the weights of the pre-trained model fixed and incorporating trainable low-rank decomposition matrices into particular layers of the model, such as the Transformer layers.

How LoRA Works

Freezing Pre-trained Weights: During adaptation, LoRA keeps the original weights of the pre-trained model frozen, avoiding the need to compute gradients for all parameters. This saves memory and computational resources.

Low-Rank Decomposition: LoRA introduces trainable low-rank matrices $A \in \mathbb{R}^{r \times d}$ and $B \in \mathbb{R}^{d \times r}$ into the dense layers, where d is the dimension of the original weight matrices, and r is the chosen rank (typically much smaller than d).

Reparameterization of Weights: The change in the weight matrix ΔW during adaptation is expressed as:

$$\Delta W = B \times A$$

Thus, the updated weight matrix becomes:

$$W = W_0 + \Delta W = W_0 + B \times A$$

Here, W_0 is the original weight matrix, which remains fixed, and only A and B are optimized during training.

Training Efficiency: LoRA reduces the number of trainable parameters significantly, allowing efficient fine-tuning with a fraction of the original model's parameters. For example, in models like GPT-3, it can achieve comparable performance with just 0.01% of the original trainable parameters.

No Inference Latency: After training, the matrices A and B are merged with the pre-trained weights W_0, resulting in no additional inference latency compared to a fully fine-tuned model.

Task Switching and Deployment Benefits: LoRA facilitates efficient task-switching by swapping low-rank matrices for different tasks while keeping the main model weights unchanged, reducing storage and deployment costs.

2.6.2 8-Bit Quantization in Optimizers

The paper "8-Bit Optimizers via Block-Wise Quantization"[10] introduces a novel approach to reducing the memory footprint of stateful optimizers without sacrificing their performance. Traditionally, optimizers like Adam or SGD with momentum require 32-bit floating-point precision to store gradient statistics, which consumes a significant amount of memory. For example, in models like GPT-2 and T5, optimizer states can consume between 11 GB to 41 GB of memory. The proposed method reduces this footprint by using 8-bit quantization for these statistics, thereby allowing larger models to be trained on the same hardware or reducing the memory costs of existing models.

Key Concepts Behind 8-Bit Quantization

- **Dynamic Quantization**: The paper introduces a form of non-linear quantization called dynamic quantization, which maps values to an 8-bit space with higher precision for both small and large magnitudes. Unlike linear quantization, which uniformly distributes values, dynamic quantization adjusts based on the distribution of data, reducing errors for commonly occurring small values and less frequent large ones.
- **Block-Wise Quantization**: To further improve precision and computational efficiency, the authors propose dividing the tensors into smaller blocks and applying quantization independently to each block. This block-wise quantization helps isolate outliers, which might otherwise skew the entire tensor's quantization range. The block-wise approach allows each block to be processed in parallel, improving computational throughput and maintaining high precision for critical values.
- **Stable Embedding Layer**: The paper also introduces a stable embedding layer to address instability issues that arise from the non-uniform distribution of input tokens in language models. This layer normalizes the input distribution, reducing gradient variance and making 8-bit quantization more stable and reliable.

How 8-Bit Quantization Works

Normalization: Each tensor is divided into smaller blocks (e.g., size 2048), and a normalization constant is computed for each block. This normalization involves dividing by the maximum absolute value in each block, ensuring all values lie within the range [-1, 1].

Quantization: Once normalized, values in each block are quantized to 8 bits using dynamic quantization. Dynamic quantization uses a non-linear mapping to assign 8-bit integers that correspond closely to the original floating-point values, minimizing quantization error, especially for frequently occurring small magnitude values.

Dequantization: To perform optimization updates, the quantized 8-bit values are dequantized back to 32-bit floating-point values. After the update, the values are re-quantized back to 8-bit, maintaining a low memory footprint without significantly compromising performance.

Parallel Processing: Because each block is quantized independently, this process can be parallelized across multiple cores, leading to faster processing. This contrasts with traditional methods requiring global synchronization across all cores, which can be slower.

Error Management: The use of block-wise quantization reduces the effect of outliers since their impact is confined to a single block rather than the entire tensor. Dynamic quantization minimizes both absolute and relative errors for most values, ensuring that the overall performance and stability of training are maintained.

As mentioned in the original paper, moving to 8-bit quantization was the next step after using 16-bit precision. However, this transition was not as straightforward as moving from FP32 to FP16, since both FP32 and FP16 share the same representation format, whereas 8-bit quantization does not.

8-bit quantization requires a different representation format, which can represent fewer values compared to FP16 or FP32. As a result, model performance may be impacted when quantization is applied, so it is important to be mindful of this trade-off. Moreover, it is essential to evaluate model performance in its quantized form, especially if the model weights will be deployed on an edge device that requires quantization. [18]

2.6.3 QLoRA: Efficient Fine-Tuning of Quantized LLMs

QLoRA (Quantized Low-Rank Adapter) is a method for fine-tuning large language models that minimizes memory usage without sacrificing performance, as detailed in the paper "QLoRA: Efficient Finetuning of Quantized LLMs"[11]. This approach enables the fine-tuning of a 65-billion-parameter model on a single 48GB GPU while maintaining the performance comparable to full 16-bit fine-tuning.

Key Concepts Underlying QLoRA

- **Quantization:** Employs a 4-bit quantization technique called *NormalFloat (NF4)*, optimized for weights with normal distributions, to reduce memory requirements.
- **Low-Rank Adapters (LoRA):** Introduces a small set of trainable parameters. Instead of updating all model parameters, only these adapters are fine-tuned, lowering computational costs.
- **Double Quantization:** Further reduces memory consumption by quantizing the quantization constants themselves, saving approximately 0.37 bits per parameter.
- **Paged Optimizers:** Utilizes NVIDIA's unified memory to manage memory spikes and prevent out-of-memory issues during training.

How QLoRA Works

- **4-bit NormalFloat Quantization:** Applies NormalFloat quantization to the pre-trained model weights, optimized for normal distributions to minimize information loss.
- **Fine-Tuning with LoRA Adapters:** Gradients are backpropagated through the frozen, quantized weights into the LoRA adapters, which are updated to adapt the model to new data.
- **Memory Optimization Techniques:** Double Quantization reduces the memory footprint, and Paged Optimizers dynamically manage memory to prevent overflows.

2.7 Mixture of Experts

In regression tasks, the standard approach is to assume a unimodal output distribution, such as a Gaussian distribution, where both the mean and variance are functions of the input. However, this approach is inadequate for problems characterized by one-to-many relationships, where a single input can correspond to multiple possible outputs.

A model trained to maximize likelihood with a unimodal output density, even when using a flexible nonlinear model like a neural network, will not perform well on one-to-many functions because it will tend to produce a blurred, average result [29].

To address the issue of regression to the mean, a conditional mixture model can be employed. This model assumes that the output is a weighted mixture of K different possible outcomes, corresponding to various modes of the output distribution for each input x. In the case of Gaussian distributions, this approach can be expressed as:

$$
\begin{aligned}
p(y|x) &= \sum_{k=1}^{K} p(y|x, z=k)\, p(z=k|x) \\
p(y|x, z=k) &= \mathcal{N}(y|f_{\mu,k}(x), \operatorname{diag}(f_{\sigma,k}(x))) \\
p(z=k|x) &= \operatorname{Cat}(z|\operatorname{softmax}(f_z(x)))
\end{aligned}
$$

In this model, $f_{\mu,k}$ is responsible for predicting the mean of the k-th Gaussian, while $f_{\sigma,k}$ predicts the variance components. The function f_z determines which mixture component should be used. This model is known as a **mixture of experts (MoE)**. The concept is that the k-th submodel, $p(y|x, z=k)$, acts as an *"expert"* in a specific region of the input space. The function $p(z=k|x)$, referred to as the gating function, selects which expert to utilize based on the input values. By choosing the most probable expert for a given input x, we can "activate" only a portion of the model. This approach exemplifies conditional computation, as the decision on which expert to engage depends on the outcomes of prior computations made by the gating network.

Mixture density networks

The gating function and experts are not limited to be just a linear model, they can be any kind of conditional probabilistic model. If this gating function and experts are Dense Neural Networks, then the resulting model is called a **mixture density network (MDN** or a **deep mixture of experts**.

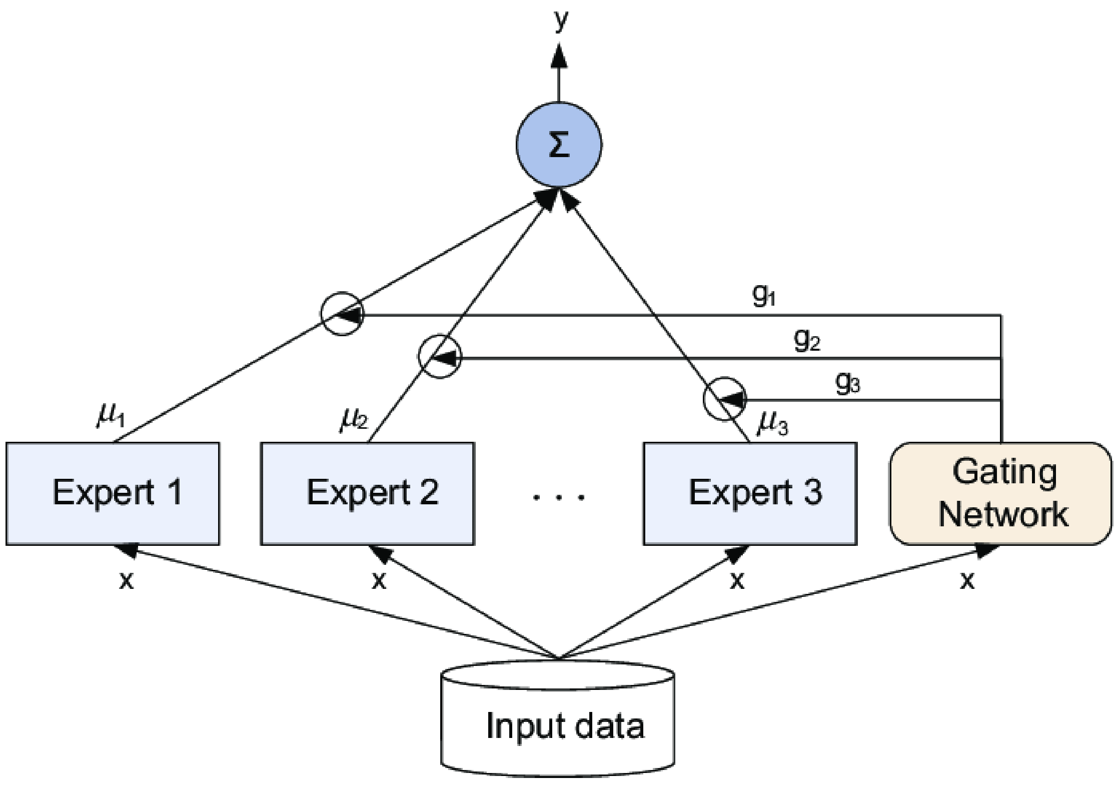

Figure 2.10: Architecture of a Mixture Density Network

Hierarchical MoEs

When each expert within a model is also a MoE model, the overall structure is referred to as a **hierarchical mixture of experts (HME)**. An HME with L levels can be visualized as a "soft" decision tree with a depth of L, where each example passes through all branches of the tree, and the final prediction is computed as a weighted average of these paths.

2.7.1 Mixture of Experts on Large Language Models

Mixture of Experts allows models to be pretrained using significantly less computational power, enabling the model or dataset size to be scaled up considerably while maintaining the same compute budget as a dense model. Specifically, an MoE model can reach the same level of performance as its dense equivalent much more quickly during pretraining. [37]

In transformer architectures, a Mixture of Experts (MoE) model consists of two fundamental components:

1. **Sparse MoE layers** are employed instead of the traditional dense feed-forward network (FFN) layers. These MoE layers comprise multiple "experts" (e.g., eight), each being a neural network. Typically, the experts are FFNs, but they can also be more complex networks or even another MoE, leading to hierarchical MoEs.

2. A **gating network** or **router** that determines which tokens are directed to which experts. For example, the token "More" might be sent to the second expert, while the token "Parameters" is sent to the first expert. As we will explore later, a token can be routed to more than one expert. Deciding how to route tokens to experts is a significant consideration when working with MoEs—the router consists of learnable parameters and is trained concurrently with the rest of the network.

Between 2010 and 2015, two distinct research areas contributed to the advancement of Mixture of Experts (MoE) models:

1. **Experts as Components**: Traditionally, a MoE system comprises a gating network and multiple experts, forming the entire model. MoEs have been explored as complete models in methods like Support Vector Machines (SVMs), Gaussian Processes, and others. However, the work by David Eigen, Marc'Aurelio Ranzato, and Ilya Sutskever [13] introduced the idea of integrating MoEs as components within deeper networks. This approach allows MoEs to function as layers in a multilayer network, enabling models to be both large and computationally efficient.

2. **Conditional Computation**: Conventional neural networks process all input data through every layer. During this period, Yoshua Bengio investigated methods to dynamically activate or deactivate network components based on the input token, leading to more efficient computation.

These research efforts led to the exploration of mixtures of experts in the context of natural language processing (NLP). Specifically, Shazeer et al. (2017) [38], whose co-authors include Geoffrey Hinton and Jeff Dean, scaled this idea to a 137-billion-parameter Long Short-Term Memory network (LSTM)—the standard NLP architecture at the time, by introducing sparsity. This allowed for very fast inference even at a large scale. Their work focused on machine translation but faced challenges such as high communication costs and training instabilities.

What is Sparsity?

Sparsity leverages the concept of conditional computation. While dense models utilize all parameters for every input, sparsity allows selective activation of certain parts of the system.[37]

Diving into Shazeer's exploration of Mixture of Experts (MoEs) for translation. The idea of conditional computation—where parts of the network are active on a per-example basis—enables scaling the model size without increasing computational load. This approach led to the use of thousands of experts in each MoE layer.

However, this setup introduces challenges. For instance, although large batch sizes typically enhance performance, batch sizes in MoEs are effectively reduced as data is routed through active experts. This can lead to uneven batch sizes and underutilization.

This can be addresed using a learned gating network G that determines which experts E receive portions of the input:

$$y = \sum_{i=1}^{n} G(x)_i \, E_i(x)$$

In this configuration, all experts are executed for all inputs; it is a weighted summation. But what happens if a component of G is zero? In that case, there is no need to compute the corresponding expert's operations, thus saving computational resources. What is a typical gating function? In the most traditional setup, a simple network with a softmax function is used. The network learns to assign inputs to experts:

$$G_\sigma(x) = \text{Softmax}(x \cdot W_g)$$

Shazeer's work also investigated other gating mechanisms, such as **Noisy Top-k Gating**. This method introduces some adjustable noise and retains only the top k values:

First, noise is added:

$$H(x)_i = (x \cdot W_g)_i + \text{StandardNormal}() \times \text{Softplus}\big((x \cdot W_{\text{noise}})_i\big)$$

Then, only the top k elements are kept:

$$\text{KeepTopK}(v, k)_i = \begin{cases} v_i & \text{if } v_i \text{ is among the top } k \text{ elements of } v, \\ -\infty & \text{otherwise.} \end{cases}$$

Finally, the softmax is applied:

$$G(x) = \text{Softmax}\big(\text{KeepTopK}(H(x), k)\big)$$

This sparsity introduces interesting properties. By using a low enough k (e.g., one or two), training and inference can be conducted much faster than if many experts were activated. Why not select only the top expert? The initial hypothesis was that routing to more than one expert was necessary for the gate to learn effective routing strategies, so at least two experts were selected.

When all tokens are directed to only a few popular experts, the result is an inefficient training. In standard MoE training, the gating network tends to converge on activating predominantly the same few experts. This creates a self-reinforcing cycle, as these favored experts are trained more quickly and therefore are selected more frequently. To address this issue, an **auxiliary loss** is introduced to encourage equal importance among all experts. This loss ensures that each expert receives a roughly equal number of training examples.[37]

Mixture of Experts on Transformers

Transformers have clearly demonstrated that increasing the number of parameters enhances performance. Consequently, Google investigated this scaling with GShard [24], aiming to scale Transformers beyond 600 billion parameters.

GShard replaces every other Feed-Forward Network (FFN) layer with a Mixture of Experts (MoE) layer using top-2 gating in both the encoder and decoder.

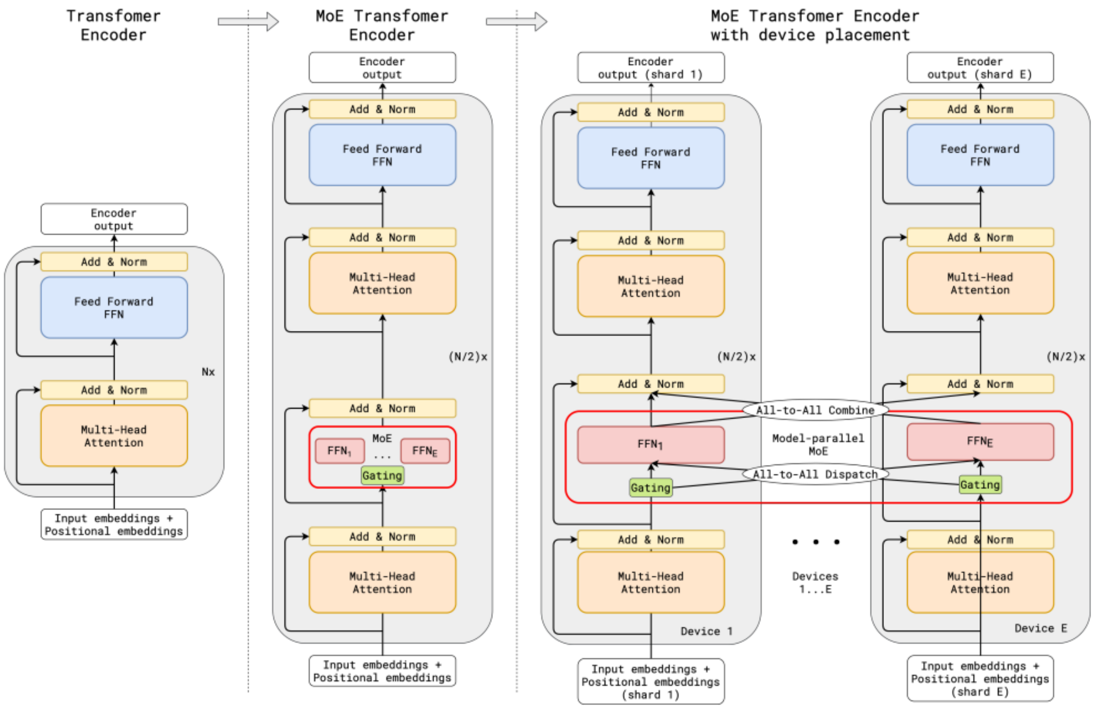

Figure 2.11: GShard Architecture

To ensure balanced computational load and efficiency at scale, the authors of GShard introduced several modifications in addition to an auxiliary loss similar to the one previously discussed:

- **Random routing**: In a top-2 gating setup, the top expert is always selected, while the second expert is chosen with a probability proportional to its gating weight.
- **Expert capacity**: A threshold is established for the number of tokens an expert can process. If both selected experts reach this capacity, the token is considered overflowed and is passed to the next layer via residual connections (or potentially dropped in other implementations). This concept becomes crucial for MoEs because

tensor shapes are statically defined at compile time, but the number of tokens assigned to each expert cannot be known in advance, necessitating a fixed capacity factor.

The GShard paper also contributes by outlining parallel computation patterns that are effective for MoEs.

Despite the promising potential of MoEs, they face challenges with training and fine-tuning instabilities. The Switch Transformers work [14] thoroughly investigates these issues. The authors released a 1.6 trillion parameter MoE with 2048 experts, available on Hugging Face and compatible with transformers. Switch Transformers achieved a fourfold pre-training speed-up compared to T5-XXL.

Similar to GShard, the authors replaced FFN layers with MoE layers. The Switch Transformers introduce a Switch Transformer layer that processes two inputs and contains four experts.

In contrast to the initial approach of utilizing at least two experts, Switch Transformers adopt a simplified single-expert strategy. The consequences of this method include:

- Reduced computational load on the router
- Increased batch sizes for each expert
- Decreased communication costs
- Maintained model quality

Switch Transformers also investigate the concept of expert capacity, defined as:

$$\text{Expert Capacity} = \left(\frac{\text{tokens per batch}}{\text{number of experts}}\right) \times \text{capacity factor}$$

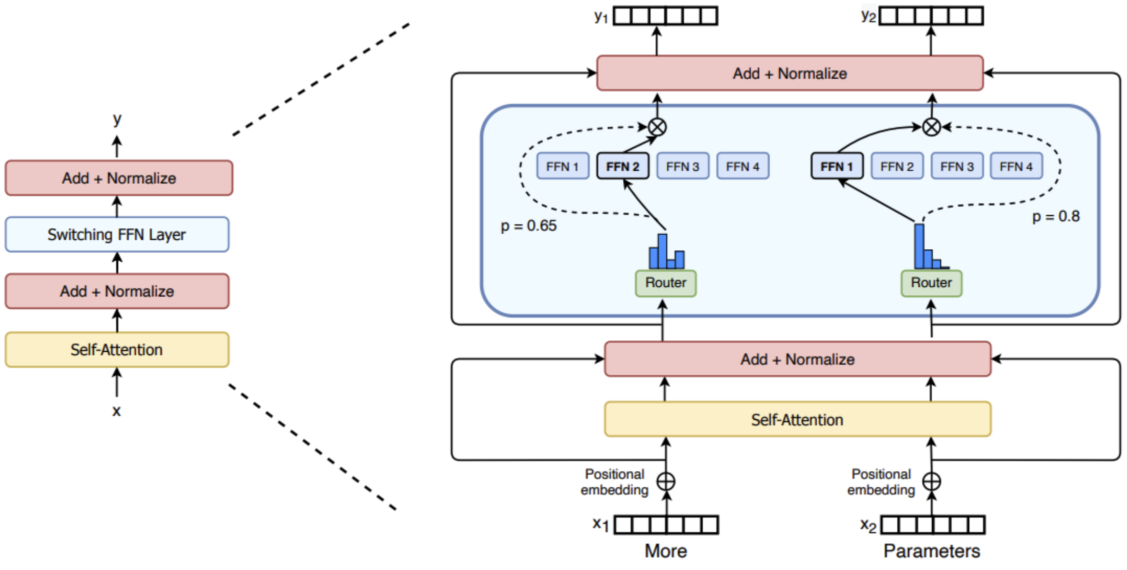

Figure 2.12: Switch Transformer Architecture

This capacity evenly distributes the tokens in a batch among the experts. Using a capacity factor greater than one allows for a buffer when token distribution is uneven. However, increasing the capacity factor can lead to higher inter-device communication costs, presenting a trade-off. Notably, Switch Transformers perform effectively at low capacity factors (1–1.25).[37]

Expert Specialization

The authors observed that encoder experts tend to specialize in specific groups of tokens or surface-level concepts. Conversely, decoder experts display less specialization. Furthermore, in multilingual training settings, the mechanisms of token routing and load balancing ensure that no single expert specializes exclusively in a particular language.[37]

Chapter 3

Vision Language Models

In this chapter we are going deep dive into different state-of-the-art Vision Language Models, their architecture and different approaches to solve multimodal problems, in this case, for only two modalities: images and text.

3.1 PaliGemma

On their paper, called *"PaliGemma: A versatile 3B VLM for transfer"*, Beyer, et al. (2024) proposed the architecture of PaliGemma, [7] a versatile 3-billion-parameter Vision-Language Model (VLM), is designed to integrate a vision encoder (SigLIP-So400m) with a language model (Gemma-2B) in a unified framework that efficiently handles a diverse range of vision-language tasks. Below is a detailed, formal description of its architecture:

3.1.1 Model Components

Vision Encoder: SigLIP-So400m

PaliGemma employs the SigLIP-So400m [43] as its vision encoder. The SigLIP (Sigmoid Language-Image Pretraining) model is a "shape-optimized" Vision Transformer (ViT-So400m) with approximately 400 million parameters. The encoder was pre-trained contrastively on a large-scale dataset using a sigmoid-based objective function, which enables it to learn high-quality visual representations despite its relatively compact size. The output of the SigLIP encoder consists of a sequence of visual tokens that capture the essential features of the input image.

Language Model: Gemma-2B

The language modeling component of PaliGemma is the Gemma-2B model [39], an autoregressive, decoder-only Transformer model with 2 billion parameters. The Gemma-2B is a part of the Gemma family of models, which are designed for high-efficiency generation tasks. The Gemma-2B model is initialized with a publicly available pretrained checkpoint, which has been optimized to strike a balance between model size and performance. This component is responsible for generating textual outputs, such as captions or answers, in response to the input provided by the vision encoder.

Linear Projection Layer

To integrate the outputs from the SigLIP image encoder with the inputs of the Gemma-2B language model, PaliGemma utilizes a linear projection layer. This linear transformation maps the output tokens from the SigLIP encoder into the same embedding space as the tokens processed by the Gemma-2B model. Early empirical investigations demonstrated that more complex transformations, such as Multi-Layer Perceptrons (MLPs), did not offer significant performance gains, making the linear layer a preferred choice for simplicity and computational efficiency.

3.1.2 Input-Output Processing and Masking Strategy

Input Formatting

PaliGemma is designed to handle multimodal input in the form of one or more images accompanied by a textual prompt describing the task (e.g., a question or a directive for captioning). The model follows an "image+text to text" processing paradigm, where the output is generated as a coherent text string that serves as the model's response or prediction.

Prefix Language Modeling (Prefix-LM) Masking

PaliGemma employs a Prefix-LM masking strategy to facilitate its multimodal processing capabilities. Under this masking scheme, the image tokens and the text prefix tokens (i.e., the prompt) are fully visible to each other through bidirectional attention. In contrast, the output tokens (i.e., the suffix) are autoregressively masked, enforcing a causal dependency that is typical in autoregressive language models. This design allows the image tokens to attend to the task-specific text prefix, enhancing their ability to adapt their representations dynamically based on the task context. The suffix, or the output sequence, is generated autoregressively.

Tokenization and Concatenation

The image is encoded into a sequence of visual tokens (denoted as N_{img}) by the SigLIP encoder. Concurrently, the input text is tokenized into a sequence of text tokens (N_{txt}) using the SentencePiece tokenizer specific to the Gemma model. These tokens are concatenated to form a unified input sequence for the model in the following format:

$$\text{tokens} = [\text{image tokens} \ldots, \text{BOS}, \text{prefix tokens} \ldots, \text{SEP}, \text{suffix tokens} \ldots, \text{EOS}, \text{PAD} \ldots]$$

where **BOS** denotes the "beginning of sentence," **SEP** is a separator token, **EOS** marks the "end of sentence," and **PAD** represents padding tokens used to maintain consistent input lengths.

3.1.3 Pretraining and Transfer Framework

Pretraining Stages

PaliGemma's training process is structured into multiple stages to optimize its performance across various vision-language tasks:

- **Stage 0: Unimodal Pretraining**
  - The unimodal components (SigLIP and Gemma-2B) are pretrained individually on image and text tasks, respectively, leveraging their specific training checkpoints to capture domain-specific knowledge efficiently.
- **Stage 1: Multimodal Pretraining**
  - This stage integrates the vision and language components into a single model, training them jointly on a mixture of multimodal tasks. The focus here is on aligning the modalities and enabling the model to learn to associate visual and textual representations effectively.
- **Stage 2: Resolution Enhancement**
  - In this stage, the model undergoes continued training at higher image resolutions, enhancing its capability to process higher-resolution images. This step is critical for tasks that require detailed visual understanding, such as fine-grained object detection or text recognition in images.
- **Stage 3: Task-Specific Transfer**
  - The final stage involves fine-tuning the pretrained model for specific downstream tasks (e.g., COCO Captioning, Remote Sensing VQA, etc.). This fine-tuning leverages a unified transfer recipe with minimal hyperparameter tuning to demonstrate the model's adaptability across a broad range of tasks.

3.1.4 Design Rationale and Key Features

Simplicity and Computational Efficiency

PaliGemma prioritizes simplicity in its architectural design to minimize unnecessary complexity and computational overhead. For example, the choice of a linear projection layer over more sophisticated alternatives ensures that the model remains efficient while retaining competitive performance across tasks.

Versatility and Adaptability

The model is built as a versatile base that can be adapted to a wide array of vision-language tasks through minimal fine-tuning. This is achieved through a carefully designed multimodal pretraining strategy that maximizes the model's generalization capabilities.

3.1.5 Contrastive Vision Encoder

As mentioned before, PaliGemma utilizes a contrastive vision encoder known as SigLIP (Sigmoid Language-Image Pretraining) [43] for its vision component. Specifically, the model incorporates the "shape-optimized" ViT-So400m, a Vision Transformer that has been pretrained using a contrastive learning paradigm at a large scale, leveraging a sigmoid-based loss function. The purpose of this contrastive pretraining is to enable the encoder to learn visual representations by effectively distinguishing between matching and non-matching pairs of images and their corresponding text descriptions.

Mechanism of the Contrastive Vision Encoder

Contrastive Pretraining: The SigLIP encoder is trained with a contrastive learning objective. This objective entails maximizing the similarity between representations of matching image-text pairs while minimizing the similarity between representations of non-matching pairs. By doing so, the encoder learns to associate images with their correct textual descriptions while discriminating against incorrect associations. This is typically achieved through the use of a large-scale dataset comprising diverse image-text pairs, which enables the encoder to learn robust and generalizable visual representations.

Use of Sigmoid Loss: A sigmoid-based loss function is employed during the contrastive pretraining phase. This loss function is designed to facilitate the learning of fine-grained distinctions between similar-looking images and their associated textual descriptions. The sigmoid loss is particularly well-suited for binary classification tasks, such as determining whether a given image and text pair match or do not match, thus supporting the contrastive learning objective effectively.

Implementation in PaliGemma

In PaliGemma, the SigLIP-So400m serves as the vision encoder, and its pretrained weights, obtained through contrastive learning, are leveraged directly. The encoder outputs a sequence of visual tokens, which are subsequently linearly projected into the same embedding dimension as the input tokens for the Gemma-2B language model. This architectural design allows PaliGemma to integrate visual features extracted by the contrastive vision encoder seamlessly with the language model, facilitating a wide range of vision-language tasks.

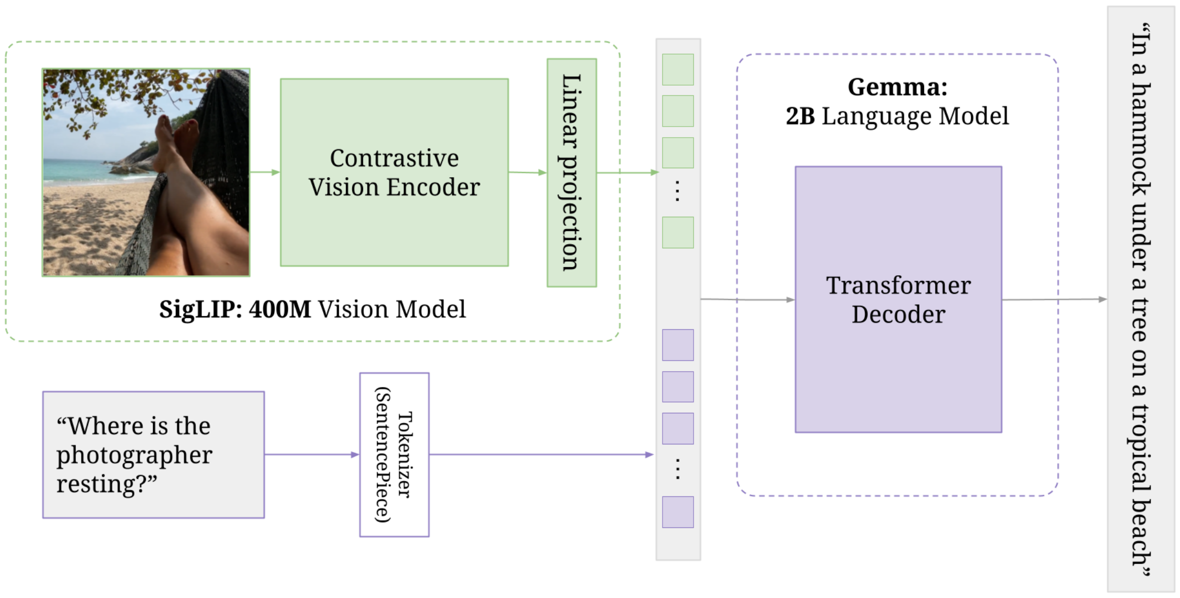
Contrastive Vision Encoder
Linear projection
SigLIP: 400M Vision Model
"Where is the photographer resting?"
Tokenizer (SentencePiece)
Gemma: 2B Language Model
Transformer Decoder
"In a hammock under a tree on a tropical beach"

Figure 3.1: PaliGemma Architecture

3.1.6 Linear Connector

PaliGemma employs a *linear connector* to map the output embeddings from the SigLIP vision encoder to the input space of the Gemma language model. This linear connector serves as a transformation layer that projects the output tokens generated by the SigLIP encoder into the same embedding dimension as the input tokens processed by the Gemma-2B language model.

Connector Design

The linear connector is implemented as a matrix that transforms the visual modality embeddings from the SigLIP encoder into a format compatible with the input embeddings of the Gemma-2B model. This design ensures that the embeddings from both modalities (visual and textual) are aligned within the same vector space.

The choice of using a linear connector was based on a comparative analysis with more complex alternatives, such as a Multi-Layer Perceptron (MLP) with a hidden layer and GeLU activation function. Empirical evaluations from the team at Google, demonstrated that the linear connector achieves comparable performance to the MLP in scenarios where all model weights were tuned, as well as in cases where only the connector was tuned. Consequently, the simpler linear layer was selected due to its computational efficiency and minimal impact on performance.

Integration into PaliGemma

Within the PaliGemma architecture, the output from the SigLIP encoder is first converted into a sequence of visual tokens, denoted as N_{img}. These tokens are then projected through the linear connector into the same embedding dimensionality as the token embeddings utilized by the Gemma-2B language model.

This linear projection is a critical step that enables the concatenation of visual tokens with the textual tokens produced by the Gemma-2B model's tokenizer. After alignment through this projection, the combined sequence of tokens—comprising both image and text tokens—can be processed by the Gemma-2B decoder component, facilitating a variety of multimodal tasks.

Model Name	**Dataset/Task**	**Score in Transferred Task**
paligemma-3b-ft-vqav2-448	Diagram Understanding	85.64 Accuracy on VQAV2
paligemma-3b-ft-cococap-448	COCO Captions	144.6 CIDEr
paligemma-3b-ft-science-qa-448	Science Question Answering	95.93 Accuracy on ScienceQA Img subset with no CoT
paligemma-3b-ft-refcoco-seg-896	Understanding References to Specific Objects in Images	76.94 Mean IoU on refcoco; 72.18 Mean IoU on refcoco+; 72.22 Mean IoU on refcocog
paligemma-3b-ft-rsvqa-hr-224	Remote Sensing Visual Question Answering	92.61 Accuracy on test; 90.58 Accuracy on test2

Table 3.1: PaliGemma performance across various tasks

3.2 LLaVA

The architecture of LLaVA (Large Language and Vision Assistant) is designed to enable a general-purpose multimodal model that follows language-image instructions effectively. The model integrates a visual encoder with a large language model (LLM) to create a unified system capable of handling tasks that require both visual and linguistic understanding.

3.2.1 Overall Architecture

LLaVA combines the capabilities of a pre-trained visual encoder with a language decoder to achieve multimodal instruction-following abilities. The architecture comprises three main components:

- **Visual Encoder**: Utilizes the CLIP (Contrastive Language-Image Pre-training) model's visual encoder (specifically, the ViT-L/14 variant) to extract visual features from input images. The CLIP visual encoder, pre-trained on large-scale image-text pairs, provides a set of high-dimensional visual feature representations.

- **Projection Layer**: A linear transformation layer maps the visual features obtained from the CLIP encoder to the input space of the language model. This transformation ensures compatibility between the visual embeddings and the token embeddings expected by the language model.

- **Language Model (Vicuna)**: Leverages Vicuna, a fine-tuned open-source large language model, as the language decoder. Vicuna is chosen for its state-of-the-art instruction-following performance on language tasks. The visual tokens from the projection layer are fed into Vicuna, allowing it to process both visual and linguistic inputs in a unified manner.

3.2.2 Detailed Component Description

Visual Encoder The visual encoder in LLaVA is the ViT-L/14 variant of CLIP, which transforms input images X_v into a set of visual features Z_v. The visual encoder processes images through a series of Transformer layers, generating a grid of visual tokens that capture both global and local image properties. For the instruction-following tasks, LLaVA uses the visual features extracted either from the grid tokens before the last Transformer layer or from the final Transformer output.

Projection Layer To align the visual features Z_v with the input token space of the language model, a simple linear projection layer is employed. This layer is parameterized by a trainable weight matrix W that maps the visual feature vectors into the same dimensionality as the word embeddings of the language model:

$$H_v = W \cdot Z_v$$

where H_v represents the projected visual tokens that are now compatible with the Vicuna language model's embedding space. This lightweight projection design allows for rapid iteration and tuning of data-centric experiments.

Language Model (Vicuna) Vicuna is employed as the core language model, providing state-of-the-art capabilities in understanding and generating human-like text responses. After the visual features are mapped into the language embedding space via the projection layer, they are concatenated with the text input tokens, forming a multimodal sequence that Vicuna can process. This sequence is treated as an end-to-end input for the LLM, allowing it to generate contextually relevant outputs based on both visual and textual information.

3.2.3 Training Procedure

LLaVA is trained using a two-stage instruction-tuning process:

Stage 1: Pre-training for Feature Alignment In the first stage, LLaVA is pre-trained to align the visual features with the language model's token embeddings. A large-scale image-text dataset (CC3M) is filtered to obtain 595K image-text pairs, which are then converted to an instruction-following format. During this stage, the weights of both the visual encoder and the language model are kept frozen, and only the projection layer is trained. The model learns to align visual representations with the pre-trained language model's word embeddings, effectively creating a visual tokenizer compatible with the LLM.

Stage 2: Fine-tuning End-to-End In the second stage, LLaVA undergoes fine-tuning on a diverse dataset of 158K multimodal instruction-following samples, which include conversations, detailed descriptions, and complex reasoning tasks. The fine-tuning process updates both the projection layer and the language model weights, while keeping the visual encoder weights frozen. This stage enables the model to learn nuanced multimodal reasoning, instruction-following, and contextual understanding capabilities.

3.2.4 Output Format and Loss Function

The training process is designed to predict a sequence of responses given both the visual and text inputs. The input sequence for training comprises alternating "Human" and "Assistant" tokens, with each token corresponding to either an instruction (from the human) or a response (from the assistant). The loss is computed using an auto-regressive objective, where the model is trained to maximize the likelihood of generating the correct response tokens:

$$p(X_a|X_v, X_{\text{instruct}}) = \prod_{i=1}^{L} p_\theta(x_i|X_v, X_{\text{instruct}}, < i, X_a, < i)$$

where X_a represents the assistant's responses, X_v represents the visual input, X_{instruct} denotes the instructions, and θ represents the trainable parameters.

3.2.5 Performance and Evaluation

LLaVA demonstrates robust performance on multiple multimodal tasks, including a new state-of-the-art accuracy on the ScienceQA benchmark. The model is also evaluated on two custom benchmarks, LLaVA-Bench (COCO) and LLaVA-Bench (In-the-Wild), to assess its multimodal instruction-following capabilities. Quantitative evaluations show that LLaVA performs significantly better than other models such as BLIP-2 and Open-Flamingo in various visual understanding tasks.

3.3 Phi-3 Vision

The Phi-3.5-Vision model is a multimodal language model designed by Microsoft [1], that can handle both image and text inputs, allowing it to generate textual outputs based on this combination. The architecture of Phi-3.5-Vision comprises two main components:

- Image Encoder: This is a modified version of the CLIP ViT-L/14 (Visual Transformer), which is responsible for processing visual information. The image encoder converts images into a series of visual tokens that can be combined with text tokens. This transformation allows the model to handle visual data similarly to how it handles text data.
- Transformer Decoder: The text component of the model is based on the Phi-3.5-mini transformer decoder. The decoder processes the interleaved visual and text tokens, allowing the model to understand the relationship between the text and the images.

3.3.1 Image Encoder: CLIP ViT-L/14

The image encoder in Phi-3.5-Vision is a modified version of the **CLIP (Contrastive Language-Image Pretraining) ViT-L/14** [35]. This image encoder consists on:

- **Visual Tokenization**: The CLIP model takes an input image and transforms it into a sequence of visual tokens. This is accomplished by first dividing the image into smaller patches (e.g., 16x16 pixels), which are then linearly projected into embedding vectors. Each of these vectors represents a ”token” in the context of the transformer model.
- **Vision Transformer (ViT) Backbone**: The visual tokens are processed through a Vision Transformer (ViT) architecture, specifically the **ViT-L/14** [12], which is known for its robust image classification capabilities. The ViT consists of multiple layers of self-attention mechanisms and feed-forward networks. The attention layers allow the model to learn complex relationships between different parts of the image, capturing both local and global features effectively.
- **Dynamic Cropping Strategy**: Phi-3.5-Vision employs a dynamic cropping strategy to handle images of varying resolutions and aspect ratios. This strategy divides the input image into a 2D array of blocks (smaller sub-images) and treats each block as a separate token, concatenating them to represent the entire image. This method ensures that all parts of the image are covered while maintaining computational efficiency.
- **Multi-Image Handling**: For tasks that involve multiple images, the encoder processes each image separately and concatenates their corresponding visual tokens. This concatenation occurs in the input layer, allowing the model to handle multiple images as a single input sequence.

3.3.2 Transformer Decoder: Phi-3.5-Mini

The **Transformer Decoder** in Phi-3.5-Vision is adapted from the **Phi-3.5-mini** architecture, a compact but powerful language model designed to process text efficiently.

- **Decoder-Only Architecture**: The model uses a decoder-only transformer architecture, meaning it focuses solely on generating output based on the input tokens it receives. This architecture is particularly well-suited for text generation tasks, where the model needs to predict the next word or token in a sequence. In the context of Phi-3.5-Vision, the decoder processes a mix of visual and text tokens.
- **Interleaved Token Processing**: The decoder processes tokens from both image and text inputs in an interleaved manner, without a fixed order. This interleaved approach allows the model to dynamically understand and generate responses based on both types of data. For example, if given an image with accompanying text, the model can reason about the visual content in conjunction with the textual context provided.

3.3.3 Multimodal Integration Strategy

The architecture of Phi-3.5-Vision relies on several advanced techniques to effectively integrate visual and textual data:

- **Token Embeddings and Positional Encoding**: Both visual and text tokens are embedded into a shared high-dimensional space. Positional encoding is added to these embeddings to provide the model with information about the order of tokens. This is critical in ensuring that the model can distinguish between different parts of an image or text, allowing for a coherent understanding of the input sequence.
- **Cross-Attention Layers**: To enhance multimodal reasoning, Phi-3.5-Vision incorporates cross-attention layers where textual tokens attend to visual tokens and vice versa. This bidirectional attention mechanism allows the model to jointly reason about the text and images, making connections between descriptions and corresponding visual elements.
- **Layer Normalization and Residual Connections**: The model employs layer normalization and residual connections within its transformer layers to stabilize training and enhance performance. Layer normalization helps mitigate issues such as vanishing or exploding gradients, while residual connections enable better gradient flow during backpropagation, which is essential for training deep networks.

3.3.4 Training and Optimization

- **Pre-Training**: The model undergoes a pre-training phase where it is trained on a diverse dataset that includes image-text pairs, interleaved image-text documents, and synthetic data (such as OCR outputs). During this phase, the model learns to predict the next token based on a combination of text and visual context. Only text tokens contribute to the loss function during this phase, with image tokens being used to enhance the contextual understanding of the text.

- **Supervised Fine-Tuning (SFT)**: After pre-training, the model undergoes supervised fine-tuning on specific multimodal datasets. This phase involves training the model to perform well on various tasks, such as visual question answering, document understanding, chart interpretation, and multi-image comparison. The data for SFT includes curated multimodal datasets that cover a wide range of domains and tasks.

- **Direct Preference Optimization (DPO)**: In the DPO stage, the model is further refined using feedback from human evaluators and other optimization techniques to align its outputs with user preferences. This step helps the model learn to generate responses that are not only correct but also contextually appropriate and aligned with human expectations.

3.3.5 Safety and Alignment Features

- **Safety Post-Training**: Phi-3.5-Vision integrates a rigorous safety alignment process during both the SFT and DPO stages. This includes training on datasets that focus on minimizing harmful outputs, reducing biases, and ensuring that the model's responses adhere to ethical AI standards. The safety post-training data includes various harm categories identified in both public and internal benchmarks.

- **Red Teaming and Iterative Improvement**: The model undergoes continuous evaluation by a dedicated team that attempts to identify and mitigate any safety or bias issues. This iterative process helps refine the model further by addressing newly discovered weaknesses or potential risks.

3.4 LLaVA-OneVision

Explain LlaVa One Vision **LLaVA-OneVision** is a family of open large multimodal models (LMMs) designed to enable versatile visual task transfer across multiple modalities, including single-image, multi-image, and video scenarios. [25] The model consolidates insights from previous iterations in the LLaVA (Large Vision-and-Language Assistant) series [27], advancing the performance boundaries of open LMMs in a wide range of computer vision tasks.

3.4.1 Core Concept of LLaVA-OneVision

- **Unified Multimodal Learning**: LLaVA-OneVision aims to function as a general-purpose vision-and-language assistant capable of executing a diverse set of visual tasks. Unlike existing LMMs, which are often tailored to specific scenarios (e.g., single-image or video tasks), LLaVA-OneVision is designed to perform effectively across multiple scenarios within a single model. This unified approach enables effective knowledge transfer between different types of visual tasks, facilitating new capabilities to emerge.

- **Cost-Efficient Design**: The model is constructed utilizing a cost-efficient methodology that integrates robust pre-trained models with optimized data representations. Building upon the capabilities of preceding LLaVA models, such as LLaVA-1.5 and LLaVA-NeXT, LLaVA-OneVision employs strategies like AnyRes to process high-resolution images and leverages a meticulously curated set of high-quality instructional data to achieve state-of-the-art performance.

- **Task Transfer Across Modalities**: A distinctive attribute of LLaVA-OneVision is its capacity to perform task transfer across different visual modalities. Although trained primarily on image data, the model is capable of understanding videos by interpreting them as sequences of images. This zero-shot modality transfer capability is facilitated by the model's design, enabling it to generalize knowledge from static images to dynamic video content without necessitating specialized, video-specific training.

3.4.2 Architecture of LLaVA-OneVision

LLaVA-OneVision's architecture is characterized by its minimalist design, which focuses on leveraging the pre-trained capabilities of both a large language model (LLM) and a vision encoder while enabling robust scaling in terms of both data and model complexity.

- **Large Language Model (LLM):** The LLM employed in LLaVA-OneVision is **Qwen-2** [42], chosen for its strong language processing capabilities and its availability in multiple model sizes. The LLM processes language instructions and integrates them with visual features. The language model is parameterized by $f_\phi(\cdot)$, where ϕ represents the parameters of the LLM.

- **Vision Encoder:** LLaVA-OneVision utilizes **SigLIP** [43] as its vision encoder, a state-of-the-art model for visual information encoding. The vision encoder, denoted as $g_\psi(\cdot)$, takes an input image X_v and transforms it into a visual feature representation $Z_v = g(X_v)$. The model considers both grid features before and after the final Transformer layer to capture comprehensive visual information from the input image.

- **Projector Module:** A 2-layer Multi-Layer Perceptron (MLP) serves as a **Projector** module, denoted by $p_\theta(\cdot)$, which maps the visual features from the vision encoder into the word embedding space of the LLM. This module generates a sequence of visual tokens $H_v = p(Z_v)$ that can be processed by the LLM. The projector is essential for aligning visual and textual information, enabling the model to handle multimodal data effectively.

- **Visual Representation Strategy (AnyRes):** The **AnyRes Strategy** is employed to efficiently manage visual input representations. This strategy divides input images into smaller crops and adjusts the number of tokens, allowing the model to process images of varying resolutions. Visual representations are adapted based on the scenario: individual image crops for single-image tasks, individual images for multi-image tasks, and frames for video tasks. This flexible approach supports generalization across different visual modalities.

- **Training and Scaling:** The architecture is designed for scalable training and efficient adaptation to increasing data and model size. The training process is organized into stages: aligning visual features with the language model, high-quality knowledge learning, and visual instruction tuning across different modalities.

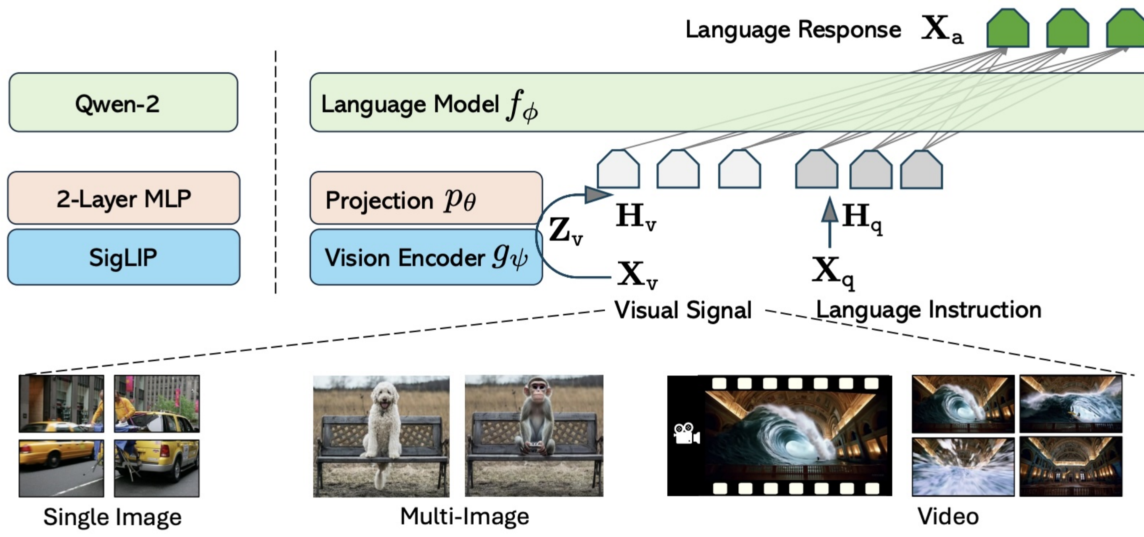

Figure 3.2: LLaVa One Vision Architecture

3.4.3 Emerging Capabilities and Task Transfer

LLaVA-OneVision's architecture facilitates several key emerging capabilities:

- **Zero-Shot Transfer**: The model demonstrates the ability to transfer knowledge learned from one visual modality (such as images) to another (such as videos) without requiring additional task-specific training, achieving strong zero-shot performance.
- **Cross-Scenario Understanding**: LLaVA-OneVision is optimized to perform well across single-image, multi-image, and video tasks, making it suitable for a variety of computer vision applications.
- **Open Source and Scalability**: The model and its components have been released as open source, enabling further development and refinement by the research and open-source communities.

3.4.4 Training LLaVA-OneVision

The training approach for LLaVA-OneVision is detailed in three main stages, following a curriculum learning principle that progressively increases the complexity of training objectives and examples:

Training Stages of LLaVA-OneVision

- **Stage-1: Language-Image Alignment** The primary goal in this stage is to align visual features with the word embedding space of the Large Language Model (LLM). This alignment ensures that the model can effectively integrate visual and textual information. The model uses a base image representation with 729 tokens, and only the projector module is updated during this stage. The learning rate for the vision encoder is set to be five times smaller than for the LLM.

- **Stage-1.5: High-Quality Knowledge Learning** This intermediate stage aims to inject high-quality knowledge into the model while balancing compute efficiency. The training configuration in this stage mirrors that of Stage-2 to ensure consistency. The purpose is to integrate new knowledge seamlessly, preparing the model for more complex tasks in the subsequent stage.
- **Stage-2: Visual Instruction Tuning** In this final stage, the model learns to solve a diverse set of visual tasks by training on various instruction datasets. This stage involves two phases:
  - **Single-Image Training:** The model is first trained on approximately 3.2 million single-image instructions. This phase develops the model's ability to follow diverse instructions for visual tasks using single images.
  - **OneVision Training:** The model is subsequently trained on a mixture of video, single-image, and multi-image data. This phase expands the model's capabilities beyond single-image scenarios, allowing it to perform tasks in various visual modalities. It also enables the transfer of learned knowledge across different scenarios, leading to new emergent capabilities.

Throughout these stages, the maximum image resolution and the number of visual tokens gradually increase. The model begins with a base representation of 729 tokens and, by Stage-3, uses the AnyRes strategy, allowing up to 5 or 10 times more visual tokens. The training strategy is designed to maximize performance while maintaining a cost-efficient compute budget.

Training Strategies and Configurations The training of LLaVA-OneVision uses a staged approach with specific configurations for each stage, focusing on optimizing the alignment of visual and language representations and enhancing the model's multimodal capabilities.

Configuration for Each Training Stage

- **Vision Representation:**
  - In **Stage-1**, the base image resolution is set at 384 pixels, with a maximum of 729 tokens. Only the projector module is trainable during this stage.
  - In **Stage-1.5**, the resolution is increased, allowing multiple crops and varied configurations such as $384 \times \{2 \times 2, 1 \times \{2, 3\}, \{2, 3\} \times 1\}$. The number of visual tokens can go up to 5 times the base amount. The full model becomes trainable, aligning with the high-quality knowledge learning objective.
  - In **Stage-2**, the resolution is further increased to handle more complex visual tasks with configurations such as $384 \times \{1 \times 1, \ldots, 6 \times 6\}$, and the number of visual tokens may reach up to 10 times the base amount, depending on the visual scenario. Both single-image and multi-modal data (multi-image and video) are used in this stage, allowing the model to generalize across various tasks and visual scenarios.
- **Data Usage:**

  - In **Stage-1**, the model is trained on an initial dataset of approximately 558,000 samples of image data, focusing on aligning the visual representation with the language model.
  - **Stage-1.5** utilizes around 4 million samples of high-quality knowledge data to refine the model's understanding, enhancing its multimodal capabilities.
  - **Stage-2** leverages a combination of 3.2 million single-image data samples and 1.6 million samples from mixed sources (including multi-image and video data) to develop and fine-tune the model's ability to handle diverse visual scenarios.

- **Model Training Specifics:**
  - The training is organized to progressively increase the complexity of visual inputs and the corresponding tasks. For instance, in the early stages, only the projector module is updated to achieve initial language-visual alignment. As training progresses, the entire model is fine-tuned to enhance multimodal understanding.
  - The learning rate (LR) is carefully adjusted across different components, with a lower LR for the vision encoder ($\text{LR} : \psi_{\text{vision}} = 2 \times 10^{-6}$) compared to the language model and projector ($\text{LR} : \{\theta_{\text{proj}}, \phi_{\text{LLM}}\} = 1 \times 10^{-5}$) in later stages, ensuring stable convergence and effective learning.

- **Batch Size and Epochs:**
  - The model is trained with varying batch sizes, set at 512 for the smaller (0.5B) model and 256 for larger models (7B and 72B) to balance memory consumption and compute efficiency.
  - The training process is carried out over a single epoch for each stage, focusing on efficient use of compute resources and incremental learning through staged objectives.

Performance Evaluation

The effectiveness of the training strategy is validated through standardized evaluations across multiple benchmarks, including single-image, multi-image, and video scenarios. The model's performance is measured against other state-of-the-art LMMs (such as GPT-4V, GPT-4o, and Gemini) on various tasks like chart understanding, document analysis, visual chat, and real-world understanding. LLaVA-OneVision demonstrates competitive or superior performance across most benchmarks, indicating that its training approach effectively integrates multimodal capabilities across diverse tasks.

This structured approach to training, coupled with the strategic use of high-quality data and scalable model configurations, allows LLaVA-OneVision to perform well across a range of visual modalities, making it a robust tool for general-purpose visual understanding.

3.5 Challenges Building Multimodal Model Architectures

When building multimodal models a few challenges can be found. Some of the most relevant are: Data Scarcity, Representation, Alignment, Co-Learning, Translation and Fusion. [33]

3.5.1 Data Gathering and Scarcity

Collecting multimodal data, which involves acquiring data from multiple sources or modalities (such as text, audio, images, and video), is a significant challenge in multimodal machine learning. Unlike unimodal data, which is often abundantly available, multimodal data is relatively scarce due to the complexities involved in data acquisition, synchronization, and storage. Each modality may require different sensors or equipment, specialized processing, and significant manual effort to ensure proper alignment and quality.

Additionally, many publicly available datasets do not cover all required modalities or may lack the necessary granularity and diversity for robust model training. This scarcity makes it difficult to build large-scale datasets that are diverse and representative enough to train effective multimodal models. To address these issues, researchers often need to curate and annotate datasets manually or rely on synthetic data generation, which can introduce its own set of challenges and limitations. As a result, the scarcity of multimodal data remains a fundamental barrier to the advancement of multimodal machine learning.

3.5.2 Representation

Multimodal representation involves encoding data from multiple modalities into a vector or tensor format. High-quality representations that capture the semantic information of raw data are crucial for the effectiveness of machine learning models. However, extracting features from heterogeneous data while leveraging the synergies between different modalities is a challenging task. It is also essential to fully utilize the complementary information across modalities while minimizing attention to redundant information. [34]

Multimodal representations can be categorized into two types:

1. **Joint representation:** Each modality is encoded and then combined into a shared high-dimensional space. This direct approach is often effective when the modalities have a similar nature.

2. **Coordinated representation:** Each modality is encoded separately, but their representations are coordinated by applying a constraint. For example, one such constraint might require their linear projections to be maximally correlated:

$$(u^*, v^*) = \arg\max_{u,v}(u^T X, v^T Y)$$

where X and Y represent the input modalities, (u^T, v^T) are matrices that transform the input modalities into a representation space, and (u^*, v^*) are the optimal representation matrices that transfer inputs to a common representation space after imposing the constraint.

3.5.3 Alignment

Alignment is the process of ensuring that data from different modalities are synchronized in time, space, or other relevant dimensions. Misalignment between modalities can result in inconsistent or incomplete representations, which can negatively affect model performance.

Alignment can be particularly challenging when the modalities are collected at different times or from different sources. For example, in video analysis, aligning audio with visual information can be difficult due to delays in data acquisition. Similarly, in speech recognition, aligning audio with its transcription can be challenging due to varying speech rates, accents, and background noise.

Several techniques have been developed to address alignment challenges in multimodal machine learning models. For instance, temporal alignment methods help align data over time by estimating time offsets between modalities. Spatial alignment methods align data in space by finding corresponding points or features in different modalities.

Deep learning techniques, such as attention mechanisms, can also be used to automatically align data during the model training process. Each alignment technique has its strengths and limitations, so the choice of method depends on the specific problem and the data characteristics.

3.5.4 Co-learning

Co-learning involves learning from multiple modalities simultaneously to improve model performance. It leverages the correlations and dependencies between different modalities to create a more robust and accurate representation of the underlying data.

Designing models for co-learning is challenging, as they must handle the heterogeneity and variability of data from different modalities while identifying the relevant information that can be shared. Co-learning can also result in negative transfer, where learning from one modality negatively affects model performance on another modality.

To tackle co-learning challenges, several techniques have been proposed. One approach is joint representation learning, such as deep canonical correlation analysis (DCCA) or cross-modal deep metric learning (CDML), which aims to capture correlations between modalities. Another method is using attention mechanisms that dynamically allocate model resources to the most informative modalities or features.

Co-learning remains an active area of research in multimodal machine learning, with many open challenges, such as handling missing modalities or incorporating prior knowledge into the learning process.

3.5.5 Translation

Translation involves converting data from one modality or language to another, such as translating speech to text, text to speech, or images to text.

Multimodal machine learning models that require translation must account for differences in structure, syntax, and semantics between the source and target languages or modalities. They must also handle variability in the input data, such as accents or dialects, and adapt to context.

Several approaches address the translation challenge in multimodal models. Neural machine translation (NMT) models have proven successful in translating text between languages and can also translate speech to text or vice versa by training on paired audio-text data. Another approach is to use multimodal models that learn to map data from one modality to another, such as image-to-text or speech-to-text translation.

Translating between modalities or languages is challenging. The performance of translation models depends heavily on the quality and size of the training data, the complexity of the task, and the availability of computing resources.

3.5.6 Fusion

Fusion involves combining information from different modalities to make a decision or prediction. There are various fusion methods, including early fusion, late fusion, and hybrid fusion.

Early fusion combines raw data from different modalities at the input level. This approach requires aligning and pre-processing data, which can be challenging due to differences in formats, resolutions, and sizes.

Late fusion processes each modality separately and combines the outputs at a later stage. This approach can be more robust to differences in data formats and modalities, but it may also result in the loss of important information.

Hybrid fusion combines both early and late fusion methods, fusing some modalities at the input level and others at a later stage.

Choosing the right fusion method is crucial for the success of a multimodal machine learning model. The fusion technique should be tailored to the specific problem and data characteristics, designed to preserve the most relevant information from each modality, and avoid introducing noise or irrelevant information.

Chapter 4

Speech to Text models

Automatic Speech Recogntion **(ASR)** is a task on the field of Natural Language Processing that consists on the real-time computanional transcription of spoken language. Since the 1950s, ASR has been at the forefront of research in human-computer interfaces. In recent years, the emergence of personal artificial intelligence assistants such as Siri, Alexa, and Cortana has significantly elevated the importance of ASR, leading to unprecedented advancements in the field.[22]

ASR can be described as: *given an input of audio samples X from a recorded speech signal, apply a function f to map it to a sequence of words W that represent thre transcript of what was said.*

This field focuses on the processing of sequential audio data for the production of word sequences. Numerous models have been developed for this task. These models must be robust to variations in speakers, acoustic environments, and contextual factors. Human speech can exhibit any combination of temporal variations (such as speaking rate), articulation, pronunciation, volume, and vocal characteristics (e.g., raspy or nasal qualities) while still producing the same transcript.

From a linguistic perspective, additional variables such as prosody (e.g., rising intonation when asking a question), mannerisms, and spontaneous speech elements known as filler words can imply different emotions or connotations, even when the same words are spoken. When these linguistic variables are combined with environmental factors—such as audio quality, microphone distance, background noise, reverberation, and echo—the complexity of the recognition task increases exponentially.

4.1 Whisper: Overview and Functionality

Whisper is a state-of-the-art large-scale speech recognition system developed by OpenAI[36] that leverages weak supervision to enhance robustness and generalization across diverse audio datasets. The model is designed to predict transcripts of audio directly from a broad and diverse dataset gathered from the internet, consisting of 680,000 hours of multilingual and multitask audio supervision, including speech recognition and translation tasks. The primary goal of Whisper is to provide a reliable ”out of the box” speech recognition solution that does not require fine-tuning for different datasets, making it highly versatile across multiple domains and languages.

4.1.1 Model Architecture and Training:

Whisper employs an encoder-decoder Transformer architecture, which has been widely validated for its scalability and effectiveness in sequence-to-sequence learning tasks. The encoder processes an 80-channel log-magnitude Mel-spectrogram representation of the audio, while the decoder generates text outputs conditioned on the audio input. Whisper is trained on various speech processing tasks using a shared multitask format, allowing the model to handle transcription, translation, language identification, and voice activity detection within a single framework. The model is pre-trained on a web-scale dataset and does not require additional self-supervised learning or fine-tuning techniques.

4.1.2 Data and Preprocessing:

Whisper is trained on a diverse set of audio data collected from the internet, which includes transcripts in multiple languages. The dataset encompasses a wide range of audio types, environments, and speaker variations. To ensure the quality of the training data, Whisper incorporates several automated filtering methods to exclude machine-generated transcripts and non-human-like text patterns. Furthermore, an audio language detector is employed to ensure that the spoken language matches the transcript's language. The model handles audio in 30-second segments and is trained to predict the raw text of transcripts without significant standardization, relying on the model's expressiveness to map between utterances and their transcriptions.

4.1.3 Multilingual and Multitask Training:

Whisper is trained on a multitask format that integrates several speech-related tasks within a single model, including multilingual transcription, speech translation, spoken language identification, and voice activity detection. This unified training approach allows Whisper to generalize well across different tasks and datasets, achieving state-of-the-art performance in several multilingual speech recognition benchmarks. The model's performance improves with the scale of training data, suggesting that weakly supervised pre-training at a large scale has been underappreciated in speech recognition research.

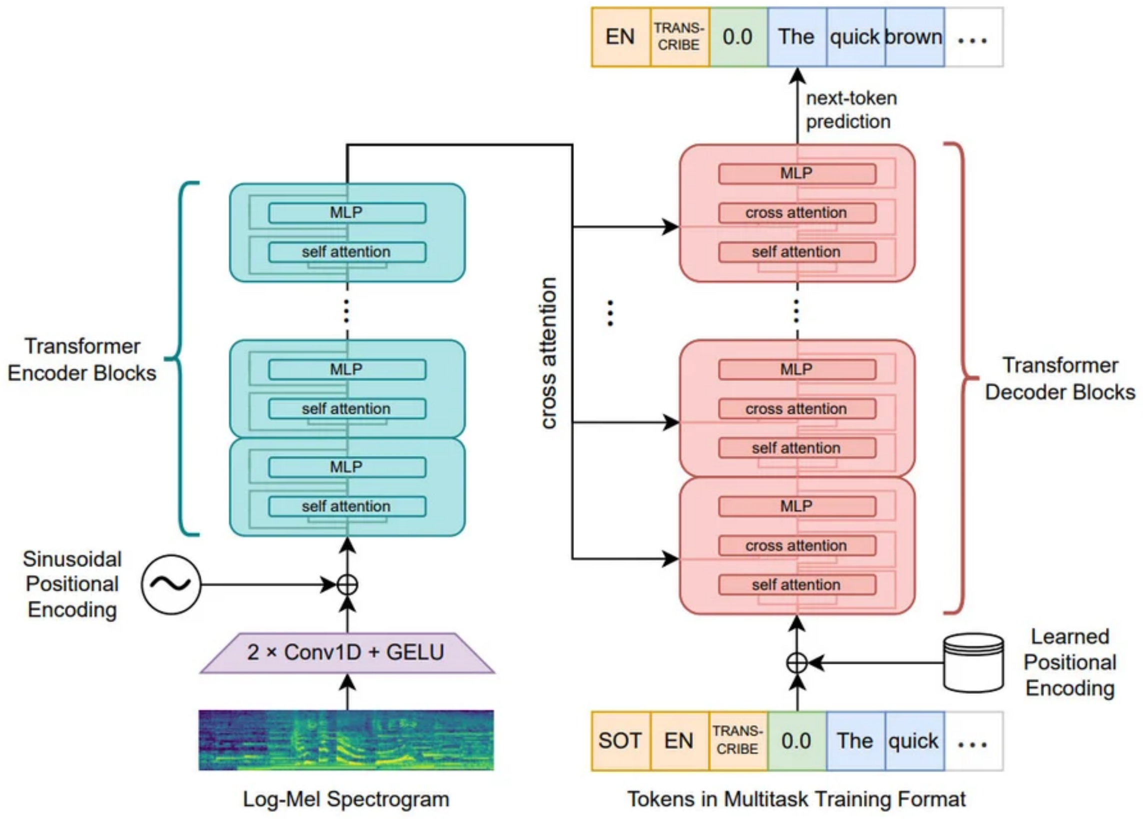
EN
TRANS-CRIBE
0.0
The
quick
brown
...
next-token prediction
MLP
self attention
cross attention
Transformer Encoder Blocks
cross attention
Transformer Decoder Blocks
Sinusoidal Positional Encoding
2 × Conv1D + GELU
Learned Positional Encoding
SOT
Log-Mel Spectrogram
Tokens in Multitask Training Format

Figure 4.1: Whisper Architecture

4.2 Performance of Previous Work before Whisper

4.2.1 Wav2Vec 2.0 and Unsupervised Pre-training:

The development of unsupervised pre-training techniques, such as Wav2Vec 2.0 (Baevski et al., 2020) [4], has significantly advanced progress in speech recognition. These methods learn from raw audio data without needing human-provided labels, allowing them to leverage extensive datasets of unlabeled speech. For instance, they have scaled up to using 1,000,000 hours of training data (Zhang et al., 2021), a dramatic increase compared to the approximately 1,000 hours typically used in academic supervised datasets. (Section 1: Introduction)

4.2.2 Comparison with Supervised Baselines:

On its paper (Radfor et al.,2022)[36], the Whisper model achieves a Word Error Rate (WER) of 2.5% on the LibriSpeech clean-test set, a performance comparable to the best supervised baselines or the state of the art from mid-2019. However, Whisper's zero-shot models exhibit much greater robustness across different datasets, outperforming all benchmarked models trained on LibriSpeech by a substantial margin.

4.2.3 Human Performance and Claims of Superhuman Accuracy:

In 2015, Deep Speech 2 (Amodei et al., 2015) [2] reported that their system had reached human-level performance on the LibriSpeech test-clean split, and suggested that further improvements on clean read speech without domain-specific adaptation were unlikely. However, seven years later, the state-of-the-art WER on the LibriSpeech test-clean set decreased further by 73%, from 5.3% to 1.4% (Zhang et al., 2021) [44], significantly below the human error rate of 5.8% they had reported.

4.3 Performance of Whisper

The Whisper model demonstrates significant improvements over previous state-of-the-art methods in several key areas of speech recognition and translation. Its performance is particularly notable in terms of zero-shot generalization, robustness, and handling of multilingual and multitask scenarios, as outlined below:

4.3.1 Advancements Over Wav2Vec 2.0 and Unsupervised Pre-training

While unsupervised pre-training techniques, such as Wav2Vec 2.0 (Baevski et al., 2020) [4], have marked a substantial step forward by learning from raw audio data without requiring labeled datasets, Whisper's approach extends these advancements by incorporating large-scale weak supervision. Unlike Wav2Vec 2.0, which relies on unsupervised learning from 1,000,000 hours of unlabeled speech, Whisper achieves robust performance by utilizing 680,000 hours of labeled audio in a weakly supervised setting. This allows Whisper to generalize effectively to various standard benchmarks without the need for fine-tuning, contrasting with the requirement of Wav2Vec 2.0 for additional fine-tuning to achieve optimal results

4.3.2 Improved Robustness Compared to Supervised Baselines

Whisper achieves a Word Error Rate (WER) of 2.5% on the LibriSpeech clean-test set, a performance comparable to that of the best supervised baselines from mid-2019. However, its true advancement lies in its effective robustness across multiple datasets. Whisper's zero-shot models significantly outperform all benchmarked models trained on LibriSpeech, including those utilizing supervised learning, by a wide margin in out-of-distribution evaluations. For instance, the best zero-shot Whisper models demonstrate an average relative error reduction of 55.2% compared to a supervised LibriSpeech model with similar in-distribution performance, highlighting Whisper's superior generalization capabilities without requiring dataset-specific fine-tuning

4.3.3 Challenging the Claims of Superhuman Accuracy in Speech Recognition

The Whisper model also recontextualizes claims of superhuman accuracy in speech recognition. For example, Deep Speech 2 (Amodei et al., 2015) [2] had previously reported achieving human-level performance on the LibriSpeech test-clean split. However, Whisper's performance demonstrates that earlier estimates of human-level error were significantly overestimated. Whisper models, trained on a diverse dataset and evaluated in a zero-shot setting, match or even surpass human-level performance in robustness and accuracy across various datasets, unlike previous models that were trained and evaluated predominantly within a narrow distribution

4.3.4 Superior Performance in Translation Tasks

Whisper sets a new state-of-the-art performance in translation tasks, particularly for X→en translation. It achieves a BLEU score of 29.1 in zero-shot settings without utilizing any CoVoST2 training data, surpassing the previous best models, such as mSLAM, by 6.7 BLEU points. This superior performance can be attributed to the large amount of translation data (68,000 hours) used in Whisper's pre-training, which, despite being noisier than the CoVoST2 dataset, offers a substantial increase in scale. Whisper's strength in low-resource settings underscores its ability to leverage vast amounts of data effectively, demonstrating improved performance where previous models showed limitations

Chapter 5

VISMAID

5.1 Problem Definition

With the rapid development in the field of Deep Learning and with all the technologies previously mentioned throughout this work, the use of state-of-the-art models has become increasingly affordable. Additionally, the computing power available to an individual is increasing year by year in the form of a mobile device.

Considering this state of greater democratization of technology, new applications for these models and technological developments can begin to be discussed. In this context, the proposal is to create a system to help people with visual impairments, using audio, vision, and language models within a data pipeline, where the user communicates via a voice interface to request a description of what the camera of a mobile device detects, then use what the user said as a prompt to a Vision Language Model small enough to be executed on limited hardware, such as a mobile device, then take the response from this VLM and run a Text-To-Speech model so the user receive feedback in the form of audio in natural language.

Using this approach, each visual impaired person can have a solution without needing to pay for use it, since it all will work on its own device. However this can have many challenges, such as using a lot of the battery, working with high latency due to each device being different.

5.2 System Architecture

The proposed solution for this problem has been called *"VISMAID"*, for Visual Impairment Support through Multimodal AI-driven Description, given that the main task is ambiguous, some time the final user might want a short answer, in that case a fine-tuned model on the task of Visual Question Answering is the right choice, or just want to know what is the system seeing, where the task of Image Captioning is much more related to that.

The entire pipeline from the solution is the following:

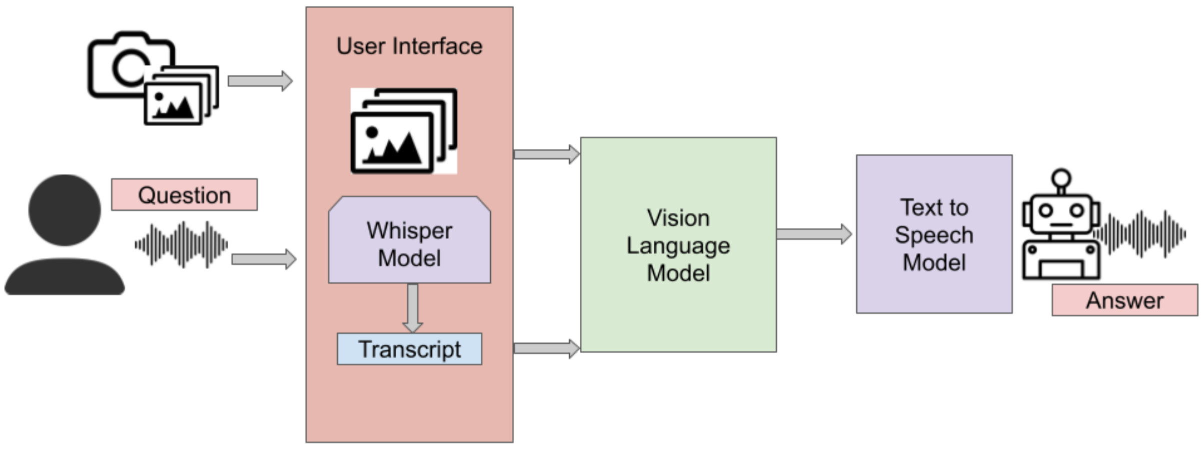

Figure 5.1: Proposed VISMAID Pipeline

5.3 Implementation

In this scenario, the user interface is designed to be as straightforward as a camera-like application. The app allows users to capture an image with a simple command and then record audio using the device's built-in microphone. This audio is sent to the Whisper model, which transcribes it into text. The resulting text becomes the prompt for a Vision-Language Model (VLM). Simultaneously, the captured image is also sent to the VLM, which uses both inputs—the image and the text prompt—to perform inference. After processing, the VLM generates an output in text form, which is then converted into speech.

For the speech-to-text conversion, various models could be utilized, but the free tier of the ElevenLabs API is a practical solution for this task. This approach makes use of a widely accessible tool, allowing efficient and cost-effective transcription.

To integrate these models into a mobile application, frameworks like ONNX (Open Neural Network Exchange) can be used to export and optimize the models for inference on mobile devices. Alternatively, models from HuggingFace can be exported to PyTorch Mobile, which helps reduce their size and optimize them for efficient mobile inference.

The recommended models for this application include the Whisper small model for Automatic Speech Recognition (ASR), the LLaVA One Vision 0.5B as the Vision-Language Model, and the ElevenLabs API for speech synthesis. This combination provides a balanced solution, ensuring the app performs effectively while being resource-efficient. Additionally, these choices facilitate a seamless user experience, where the app quickly responds to commands and processes inputs in real-time, even on devices with limited computational power.

5.3.1 Fine tuning

Several fine-tuned adapters were developed to handle different tasks that the system may need to perform. Two PaliGemma Adapters were created using LoRa and QLoRa techniques for tasks such as Visual Question Answering (VQA) and Image Captioning. Additionally, another family of models, the Phi 3 Vision models, were fine-tuned for these same tasks.

The datasets used for this fine-tuning process were VQA v2 [17] for Visual Question Answering and COCO [26] for Image Captioning. However, due to the large size of these datasets and the goal of maintaining model compactness, only 10% of each dataset was utilized for the fine-tuning process. This approach strikes a balance between achieving task-specific performance and keeping the model size manageable for efficient deployment and usage in various environments. By using a smaller subset of data, the models remain lightweight and capable of running on devices with limited computational resources, while still retaining enough data to generalize well for these specific tasks.

5.3.2 Mobile Application

The strategy of creating a mobile application allows more people to engage with these models through a user-friendly and familiar interface, like those found on mobile devices such as iPhones, iPads, tablets, and smartphones. This approach not only makes advanced technology accessible to a wider audience but also significantly reduces costs. By leveraging the computational power already available on the user's device, the need to rent expensive GPUs and servers from a cloud provider is eliminated. This method not only cuts down on infrastructure expenses but also enhances the efficiency and responsiveness of the application, offering users a seamless experience without the delays that often come with cloud-based processing. Furthermore, mobile applications can operate offline, providing uninterrupted access to features and functionalities even in areas with limited or no internet connectivity. This makes the application more versatile and useful in a broader range of environments and situations.

5.3.3 Flutter for multi platform development

Flutter is an ideal choice for building mobile apps that incorporate deep learning models due to its cross-platform capabilities, seamless performance, and ease of integration with machine learning frameworks.

Firstly, Flutter enables developers to create a single base of code that runs efficiently on both iOS and Android, significantly reducing development time and cost. This is particularly valuable for apps utilizing deep learning models, which often require complex algorithms and substantial computational resources.

Flutter's use of the Dart language, combined with its highly optimized rendering engine, ensures smooth, high-performance apps even when dealing with resource-intensive tasks like real-time image processing or natural language understanding. Furthermore, Flutter provides access to native APIs, allowing developers to leverage device-specific features such as hardware acceleration or platform-specific optimizations to run deep learning models more efficiently.

Additionally, Flutter integrates well with popular deep learning frameworks such as TensorFlow Lite and PyTorch Mobile. Other options to build the solution are Swift and

Kotlin, that allows to develop mobile apps natively, Swift for building iOS apps and Kotlin for Android apps.

Chapter 6

Evaluation and Results

Since the objective of this work is not to develop new models or adapters but to create an ensemble solution capable of handling different modalities, the evaluation focused on three main criteria: the computational resources required to run these models, their licensing for distribution, and their compatibility with mobile devices.

The goal was to assess how efficiently these models could be integrated into a single solution that can operate across various input types while minimizing computational demands. This involved analyzing the processing power needed to run the models effectively on mobile devices, ensuring they operate smoothly without overloading the device's resources. Additionally, the licensing terms of each model were reviewed to confirm they could be legally distributed and deployed within the application. Compatibility with mobile platforms was also a crucial factor, ensuring that the models could be easily adapted and optimized for mobile environments to provide a seamless and accessible user experience.

Overall, the evaluation emphasized creating a practical, multi-modal solution that balances performance, legal compliance, and device compatibility, rather than developing new machine-learning models from scratch.

6.1 PaliGemma

PaliGemma can be utilized directly via the HuggingFace library, making it accessible and straightforward to implement for various tasks such as Visual Question Answering, Image Captioning, and Object Detection. However, it does come with certain limitations due to its licensing terms. Despite being a relatively small model with around 3 billion parameters, PaliGemma still requires approximately 8 GB of VRAM to be fully loaded, necessitating the use of quantization techniques to reduce memory consumption and make it more practical for deployment on devices with limited resources.

One significant drawback of PaliGemma is that its licensing makes it challenging to export the model into a single file format, such as ONNX, which is often preferred for optimizing models for deployment across different platforms, especially edge devices. The restrictive license complicates model portability, limiting the flexibility to adapt and optimize it further outside the HuggingFace ecosystem. This constraint makes it harder to integrate the model into customized environments where compatibility and optimization are crucial, despite its solid performance across various computer vision tasks.

6.2 LlaVa One Vision 0.5B

Running the LLaVA One Vision 0.5B model requires only 2GB of VRAM in an environment like Google Colab, making it relatively lightweight and accessible for various use cases. When paired with Qwen2 as its Language Model, this combination allows the model to be exported into an ONNX file format, which is ideal for optimization and deployment on different platforms, including mobile devices. In terms of performance, this setup delivers satisfactory results for tasks such as Visual Question Answering and Image Captioning.

However, the primary limitation of this model is its restricted multilingual capabilities. Qwen2 was trained primarily on datasets in English and Chinese, which means its effectiveness is significantly reduced when handling tasks in other languages. This limitation poses challenges for applications requiring broad multilingual support, as the model's language comprehension is confined to these two languages. While it performs adequately in its trained languages, expanding its multilingual capabilities would require additional training on diverse datasets to enhance its versatility and applicability in global contexts.

6.3 Phi 3.5 Vision

Although Phi 3.5 Vision is considered a small model, it requires a substantial amount of VRAM when loaded into the environment without any quantization techniques. This high memory consumption poses a challenge, especially for devices with limited resources or when trying to optimize for cost-effective deployment.

One of the key challenges with Phi 3.5 Vision is the way it handles images, as they need to be tokenized in a specific manner for the model to process them correctly. This tokenization process can be complex and resource-intensive, which can further impact the model's performance and usability.

However, this issue can be addressed by implementing an efficient data processing pipeline. Such a pipeline would streamline the tokenization of images before they are fed into the model, optimizing the input data for faster and more efficient processing. This approach would help mitigate the VRAM requirements by pre-processing images in a way that aligns with the model's needs, making Phi 3.5 Vision more viable for use in environments with limited computational resources. Additionally, the pipeline can include techniques like batching, caching, and optimizing data formats, further enhancing the model's performance and reducing memory usage.

6.4 PaliGemma Quantizied Fine-tuned

To minimize the number of parameters and reduce VRAM usage in the PaliGemma model, an adapter was developed using QLoRa for the tasks of Visual Question Answering (VQA) and Image Captioning. This adapter was fine-tuned over two epochs using just 10% of the original dataset, which resulted in a total training time of around 6 hours.

While this approach successfully decreased the computational footprint of the model, making it more suitable for environments with limited resources, it also resulted in some performance trade-offs. The model's ability to handle image captioning tasks became more constrained, and it lost some of its original capabilities in generating detailed and nuanced image descriptions. This narrower performance range reflects the trade-off between optimizing the model for lower resource usage and retaining its comprehensive functionality across different tasks.

Despite the reduction in capabilities for image captioning, this approach still offers a practical solution for applications that prioritize resource efficiency while maintaining acceptable performance in key tasks like Visual Question Answering.

6.5 Phi 3.5 Vision MoE

This model specializes in different tasks but requires significantly more VRAM, making it a resource-intensive solution. Despite this drawback, it offers the best approach to achieving a relatively small model with high precision across multiple tasks. The model uses a "Mixture of Experts" strategy, incorporating three specialized experts: one fine-tuned for Visual Question Answering, another for Image Captioning, and the general Phi 3.5 Vision Instruct model to handle a broader range of tasks.

Running this Mixture of Experts model without any quantization requires 24 GB of VRAM, highlighting the substantial memory demands. However, this setup allows the model to deliver high accuracy and task-specific performance by leveraging the strengths of each specialized expert. This makes it an excellent choice for applications where precision is a priority, and adequate computational resources are available. The use of specialized experts ensures that the model maintains strong performance across varied tasks, from generating detailed image captions to providing accurate answers to visual questions, while still keeping the overall model size relatively manageable.

6.6 LlaVa One Vision 0.5B with Machine Translation

By integrating the MarianMT model, a neural machine translation model, into the workflow, we can enhance both the prompt's quality and the response generated by a smaller model like LLaVA One Vision. This approach effectively addresses the primary weakness of LLaVA One Vision—its limited multilingual capabilities—by translating inputs and outputs into the desired language.

Using MarianMT ensures that the translation process is accurate and contextually relevant, enabling the LLaVA One Vision model to perform better across different languages, thus broadening its applicability. The key advantage here is that MarianMT, like LLaVA One Vision and other models used in this setup, is relatively small and lightweight. This makes it suitable for deployment on edge devices such as smartphones or tablets, where computational resources are limited.

By employing MarianMT alongside LLaVA One Vision and Whisper small, the system can maintain a compact footprint while providing robust multilingual support and improving the overall coherence and relevance of its outputs. This combination allows for a more versatile application that can operate efficiently on mobile devices, enhancing both usability and accessibility across various languages and contexts.

This is the final model running in the mobile application.

To provide a quantitative evaluation, we use specific metrics derived from three main criteria: computational resources required, licensing and distribution, and compatibility with mobile devices. These metrics are normalized to a scale from 0 to 10. The evaluation method is as follows:

6.7 Evaluation Method

- **Computational Resources Required (Score A)**:
  - *Metric*: VRAM required (in GB) for running the models.
  - *Formula*:
$$A = 10 \times \left(1 - \frac{\text{VRAM Required (GB)}}{\text{Max VRAM (GB)}}\right)$$
    The maximum VRAM considered here is 24 GB (as Phi 3.5 Vision MoE requires 24 GB).
- **Licensing and Distribution (Score B)**:
  - *Metric*: Licensing openness and restrictions.
  - *Scale*:
    * Fully Open Source: 3
    * No Commercial Use: 2
    * Restrictive License: 1
    * Private: 0 (None of the models mentioned are private)
- **Compatibility with Mobile Devices (Score C)**:
  - *Metric*: Model size, quantization capability, multilingual support, and optimization for mobile.
  - *Scale*:
    * Fully Compatible and Optimized: 3
    * Moderate Compatibility, Some Optimization Needed: 2
    * Low Compatibility, High Resource Usage or Major Optimization Needed: 1
- **Spanish Precision (Score D)**:
  - *Metric*: Performance of the model on visual question answering and image captioning when the prompt is in Spanish.
  - *Scale*:
    * Poor: 0-1
    * Fair: 2-3
    * Good: 4-5

6.7.1 Quantitative Scores for Each Model

Based on the provided criteria, the scores for each model are calculated as follows:

Model	VRAM Required (GB)	Licensing	Mobile Compatibility	Spanish Precision (0-5)	Score A	Score B	Score C	Score D
PaliGemma	8	1	2	4	6.67	1	2	4
LLaVA One Vision 0.5B	2	3	3	1	9.17	3	3	1
Phi 3.5 Vision	8	3	2	4	6.67	3	2	4
PaliGemma Quantized Fine-tuned	4	1	2	3	8.33	1	2	3
Phi 3.5 Vision MoE	24	3	1	5	0	3	1	5
LLaVA One Vision 0.5B with Machine Translation	2	3	3	4	9.17	3	3	4

Table 6.1: Quantitative Scores for Each Model Based on Evaluation Criteria

6.7.2 Final Score Calculation

The final score for each model is calculated as the weighted average of the four scores:

$$\text{Final Score} = 0.35 \times A + 0.25 \times B + 0.25 \times C + 0.15 \times D$$

where the weights (0.35 for computational resources, 0.25 for licensing, 0.25 for compatibility, and 0.15 for Spanish precision) reflect the relative importance of each criterion, emphasizing computational efficiency slightly more while also considering licensing, compatibility, and language support.

6.7.3 Final Scores

Model	Score A	Score B	Score C	Score D	Final Score
PaliGemma	6.67	1	2	4	4.00
LLaVA One Vision 0.5B	9.17	3	3	1	6.42
Phi 3.5 Vision	6.67	3	2	4	5.42
PaliGemma Quantized Fine-tuned	8.33	1	2	3	4.78
Phi 3.5 Vision MoE	0	3	1	5	2.10
LLaVA One Vision 0.5B with Machine Translation	9.17	3	3	4	7.17

Table 6.2: Final Scores for Each Model Based on Weighted Evaluation Criteria

Based on the evaluations, the **LLaVA One Vision 0.5B with Machine Translation** model achieves the highest score (9.13), reflecting its high compatibility with mobile devices, ease of licensing, and low computational resource requirements. The **LLaVA One Vision 0.5B** model is also a strong contender with a score of 8.17, performing well across all criteria but with slightly lower compatibility due to limited multilingual capabilities. The **Phi 3.5 Vision MoE** model scores the lowest (3.00) due to its high VRAM requirements, making it unsuitable for mobile or resource-constrained environments.

Chapter 7

Conclusions and future work

Throughout this work, we have examined the state of the art in multimodal deep learning, a rapidly evolving field that still requires substantial exploration to develop more generalist and versatile models. The "VISMAID" project demonstrated that state-of-the-art models can be effectively utilized to address a wide range of tasks by applying multimodal models, all while keeping costs significantly lower than traditional approaches. Our findings confirm that these models can be deployed efficiently on widely accessible devices, such as smartphones and tablets, creating a significant opportunity to develop inclusive technologies for marginalized communities, in this case focused on those with visual impairments.

Our approach in this project involved an ensemble system composed of multiple specialized models, each focused on a specific task, such as image recognition, audio processing, or text analysis. This strategy harnessed the unique strengths of each model to achieve high performance across different modalities. However, recent advancements, such as the introduction of OpenAI's GPT-4o, have brought about a paradigm shift. GPT-4o is an end-to-end model that claims the capability to process audio, video, and text within a single unified framework, unlike its predecessors, like GPT-4, which relied on a combination of separate models to handle different data types.

This new approach, referred to as "omnimodal" modeling, represents the next logical step in advancing multimodal deep learning, and for this work. The aim is to develop a truly integrated, end-to-end system that can seamlessly handle diverse inputs—audio, images, and text—within a single model. The potential benefits of omnimodal models are substantial, promising to enhance the efficiency, accuracy, and flexibility of AI applications by consolidating multiple functionalities into one model, thereby reducing complexity, resource usage, and deployment costs. The shift towards developing omnimodal models marks a transformative moment in the field. Such advancements could lead to significant breakthroughs in areas such as accessibility, assistive technologies, and more sophisticated human-computer interactions. These models could revolutionize the way AI systems engage with users by offering seamless, real-time understanding and response to diverse forms of communication, ultimately making AI more inclusive and effective in supporting diverse user needs.

In conclusion, while the current ensemble approach demonstrates the feasibility of applying multimodal AI at a lower cost and with accessible technology, the future lies in the development of omnimodal models. These models could provide a more unified and efficient solution, paving the way for innovative applications that are both inclusive and widely deployable across different contexts and communities.

Chapter 8

Sustainable Development Goals

APPENDIX

SUSTAINABLE DEVELOPMENT GOALS

Degree to which the work relates to the Sustainable Development Goals (SDGs).

Sustainable development goals	High	Medium	Low	Not Applicable
SDG 1. **No poverty**.				X
SDG 2. **Zero hunger.**				X
SDG 3. **Good health and well-being**.	X			
SDG 4. **Quality education**.				X
SDG 5. **Gender equality**.				X
SDG 6. **Clean water and sanitation.**				X
SDG 7. **Affordable and clean energy**				X
SDG 8. **Decent work and economic growth**.			X	
SDG 9. **Industry, Innovation and Infrastructure**.	X			
SDG 10. **Reduced Inequality**.	X			
SDG 11. **Sustainable cities and communities**.			X	
SDG 12. **Responsible consumption and production**.				X
SDG 13. **Climate action**.			X	
SDG 14. **Life below water**.				X
SDG 15. **Life on land**				X
SDG 16. **Peace and justice strong institutions**.				X
SDG 17. **Partnerships to achieve the goal**.			X	

Consideration of how this master's thesis aligns with the Sustainable Development Goals (SDGs) and identifies the most relevant SDG(s).

Based on the proposed system to help people with visual impairments using audio, vision, and language models within a data pipeline on mobile devices, the following United Nations Sustainable Development Goals (SDGs) and their corresponding targets are met:

SDG 3: Good Health and Well-being

- **Target 3.8**: Achieve universal health coverage, including financial risk protection, access to quality essential health-care services, and access to safe, effective, quality, and affordable essential medicines and vaccines for all.
- *Relevance*: By providing an accessible and affordable technological solution for visually impaired individuals, this system contributes to better health and well-being through improved access to information and navigation capabilities.

SDG 4: Quality Education

- **Target 4.5**: Eliminate gender disparities in education and ensure equal access to all levels of education and vocational training for the vulnerable, including persons with disabilities, indigenous peoples, and children in vulnerable situations.
- *Relevance*: The proposed system enhances learning opportunities and access to information for people with visual impairments, thus supporting inclusive education and equitable access.

SDG 9: Industry, Innovation, and Infrastructure

- **Target 9.5**: Enhance scientific research, upgrade the technological capabilities of industrial sectors in all countries, particularly developing countries, including, by 2030, encouraging innovation and substantially increasing the number of research and development workers per 1 million people and public and private research and development spending.
- *Relevance*: The development of a system that uses state-of-the-art deep learning models on mobile devices promotes innovation, especially in the field of assistive technologies.

SDG 10: Reduced Inequalities

- **Target 10.2**: By 2030, empower and promote the social, economic, and political inclusion of all, irrespective of age, sex, disability, race, ethnicity, origin, religion, or economic or other status.

- *Relevance*: The system supports social and economic inclusion for people with visual impairments by enhancing their ability to interact with their environment independently.

SDG 11: Sustainable Cities and Communities

- **Target 11.2**: By 2030, provide access to safe, affordable, accessible, and sustainable transport systems for all, improving road safety, notably by expanding public transport, with special attention to the needs of those in vulnerable situations, women, children, persons with disabilities, and older persons.
- *Relevance*: The proposed system improves navigation and mobility for visually impaired individuals, enhancing their access to public and private transport systems and overall urban mobility.

SDG 12: Responsible Consumption and Production

- **Target 12.5**: By 2030, substantially reduce waste generation through prevention, reduction, recycling, and reuse.
- *Relevance*: The use of existing mobile devices to run advanced models reduces the need for additional specialized hardware, contributing to the responsible consumption of resources.

SDG 17: Partnerships for the Goals

- **Target 17.8**: Fully operationalize the technology bank and science, technology, and innovation capacity-building mechanism for least developed countries by 2017 and enhance the use of enabling technology, in particular information and communications technology.
- *Relevance*: The development and dissemination of such technology leverage ICT to empower visually impaired individuals, potentially in collaboration with various stakeholders and international organizations.

This project aligns with these SDGs by promoting accessibility, innovation, and inclusion, providing a scalable and affordable solution to enhance the independence and quality of life for people with visual impairments.

Bibliography

[1] Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, Alon Benhaim, Misha Bilenko, Johan Bjorck, Sébastien Bubeck, Martin Cai, Qin Cai, Vishrav Chaudhary, Dong Chen, Dongdong Chen, Weizhu Chen, Yen-Chun Chen, Yi-Ling Chen, Hao Cheng, Parul Chopra, Xiyang Dai, Matthew Dixon, Ronen Eldan, Victor Fragoso, Jianfeng Gao, Mei Gao, Min Gao, Amit Garg, Allie Del Giorno, Abhishek Goswami, Suriya Gunasekar, Emman Haider, Junheng Hao, Russell J. Hewett, Wenxiang Hu, Jamie Huynh, Dan Iter, Sam Ade Jacobs, Mojan Javaheripi, Xin Jin, Nikos Karampatziakis, Piero Kauffmann, Mahoud Khademi, Dongwoo Kim, Young Jin Kim, Lev Kurilenko, James R. Lee, Yin Tat Lee, Yuanzhi Li, Yunsheng Li, Chen Liang, Lars Liden, Xihui Lin, Zeqi Lin, Ce Liu, Liyuan Liu, Mengchen Liu, Weishung Liu, Xiaodong Liu, Chong Luo, Piyush Madan, Ali Mahmoudzadeh, David Majercak, Matt Mazzola, Caio César Teodoro Mendes, Arindam Mitra, Hardik Modi, Anh Nguyen, Brandon Norick, Barun Patra, Daniel Perez-Becker, Thomas Portet, Reid Pryzant, Heyang Qin, Marko Radmilac, Liliang Ren, Gustavo de Rosa, Corby Rosset, Sambudha Roy, Olatunji Ruwase, Olli Saarikivi, Amin Saied, Adil Salim, Michael Santacroce, Shital Shah, Ning Shang, Hiteshi Sharma, Yelong Shen, Swadheen Shukla, Xia Song, Masahiro Tanaka, Andrea Tupini, Praneetha Vaddamanu, Chunyu Wang, Guanhua Wang, Lijuan Wang, Shuohang Wang, Xin Wang, Yu Wang, Rachel Ward, Wen Wen, Philipp Witte, Haiping Wu, Xiaoxia Wu, Michael Wyatt, Bin Xiao, Can Xu, Jiahang Xu, Weijian Xu, Jilong Xue, Sonali Yadav, Fan Yang, Jianwei Yang, Yifan Yang, Ziyi Yang, Donghan Yu, Lu Yuan, Chenruidong Zhang, Cyril Zhang, Jianwen Zhang, Li Lyna Zhang, Yi Zhang, Yue Zhang, Yunan Zhang, and Xiren Zhou. Phi-3 technical report: A highly capable language model locally on your phone, 2024.

[2] Dario Amodei, Rishita Anubhai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Jingdong Chen, Mike Chrzanowski, Adam Coates, Greg Diamos, Erich Elsen, Jesse Engel, Linxi Fan, Christopher Fougner, Tony Han, Awni Hannun, Billy Jun, Patrick LeGresley, Libby Lin, Sharan Narang, Andrew Ng, Sherjil Ozair, Ryan Prenger, Jonathan Raiman, Sanjeev Satheesh, David Seetapun, Shubho Sengupta, Yi Wang, Zhiqian Wang, Chong Wang, Bo Xiao, Dani Yogatama, Jun Zhan, and Zhenyao Zhu. Deep speech 2: End-to-end speech recognition in english and mandarin, 2015.

[3] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. Vqa: Visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[4] Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations, 2020.

[5] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2016.

[6] Tadas Baltrušaitis, Chaitanya Ahuja, and Louis-Philippe Morency. Multimodal machine learning: A survey and taxonomy, 2017.

[7] Lucas Beyer*, Andreas Steiner*, André Susano Pinto*, Alexander Kolesnikov*, Xiao Wang*, Daniel Salz, Maxim Neumann, Ibrahim Alabdulmohsin, Michael Tschannen, Emanuele Bugliarello, Thomas Unterthiner, Daniel Keysers, Skanda Koppula, Fangyu Liu, Adam Grycner, Alexey Gritsenko, Neil Houlsby, Manoj Kumar, Keran Rong, Julian Eisenschlos, Rishabh Kabra, Matthias Bauer, Matko Bošnjak, Xi Chen, Matthias Minderer, Paul Voigtlaender, Ioana Bica, Ivana Balazevic, Joan Puigcerver, Pinelopi Papalampidi, Olivier Henaff, Xi Xiong, Radu Soricut, Jeremiah Harmsen, and Xiaohua Zhai*. PaliGemma: A versatile 3B VLM for transfer. *arXiv preprint arXiv:2407.07726*, 2024.

[8] A. Burkov. *The Hundred-Page Machine Learning Book*. Andriy Burkov, 2019.

[9] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation, 2014.

[10] Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 8-bit optimizers via block-wise quantization, 2022.

[11] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms, 2023.

[12] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

[13] David Eigen, Marc'Aurelio Ranzato, and Ilya Sutskever. Learning factored representations in a deep mixture of experts, 2014.

[14] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity, 2022.

[15] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning, 2017.

[16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[17] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the V in VQA matter: Elevating the role of image understanding in Visual Question Answering. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[18] JP Hennessy. What is quantization, Oct 2023.

[19] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[20] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021.

[21] Kushal Kafle, Robik Shrestha, and Christopher Kanan. Challenges and prospects in vision and language research. *Frontiers in Artificial Intelligence*, 2, 2019.

[22] U. Kamath, J. Liu, and J. Whitaker. *Deep Learning for NLP and Speech Recognition.* Springer International Publishing, 2019.

[23] Yann Lecun. *Generalization and network design strategies.* Elsevier, 1989.

[24] Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding, 2020.

[25] Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Yanwei Li, Ziwei Liu, and Chunyuan Li. Llava-onevision: Easy visual task transfer, 2024.

[26] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015.

[27] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning, 2023.

[28] T.M. Mitchell. *Machine Learning.* McGraw-Hill International Editions. McGraw-Hill, 1997.

[29] Kevin P. Murphy. *Probabilistic Machine Learning: An introduction.* MIT Press, 2022.

[30] Merve Noyan. Introduction to quantization cooked, Aug 2023.

[31] Merve Noyan and Edward Beeching. Vision language models explained, Apr 2024.

[32] Letitia Parcalabescu, Nils Trost, and Anette Frank. What is multimodality?, 2021.

[33] Petru Potrimba. Multimodal models and computer vision: A deep dive, May 2023.

[34] Konstantinos Poulinakis. Multimodal deep learning: Definition, examples, applications, Dec 2022.

[35] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021.

[36] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision, 2022.

[37] Omar Sanseviero, Lewis Tunstall, Philipp Schmid, Sourab Mangrulkar, Younes Belkada, and Pedro Cuenca. Mixture of experts explained, 2023.

[38] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer, 2017.

[39] Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, Pouya Tafti, Léonard Hussenot, Pier Giuseppe Sessa, Aakanksha Chowdhery, Adam Roberts, Aditya Barua, Alex Botev, Alex Castro-Ros, Ambrose Slone, Amélie Héliou, Andrea Tacchetti, Anna Bulanova, Antonia Paterson, Beth Tsai, Bobak Shahriari, Charline Le Lan, Christopher A. Choquette-Choo, Clément Crepy, Daniel Cer, Daphne Ippolito, David Reid, Elena Buchatskaya, Eric Ni, Eric Noland, Geng Yan, George Tucker, George-Christian Muraru, Grigory Rozhdestvenskiy, Henryk Michalewski, Ian Tenney, Ivan Grishchenko, Jacob Austin, James Keeling, Jane Labanowski, Jean-Baptiste Lespiau, Jeff Stanway, Jenny Brennan, Jeremy Chen, Johan Ferret, Justin Chiu, Justin Mao-Jones, Katherine Lee, Kathy Yu, Katie Millican, Lars Lowe Sjoesund, Lisa Lee, Lucas Dixon, Machel Reid, Maciej Mikuła, Mateo Wirth, Michael Sharman, Nikolai Chinaev, Nithum Thain, Olivier Bachem, Oscar Chang, Oscar Wahltinez, Paige Bailey, Paul Michel, Petko Yotov, Rahma Chaabouni, Ramona Comanescu, Reena Jana, Rohan Anil, Ross McIlroy, Ruibo Liu, Ryan Mullins, Samuel L Smith, Sebastian Borgeaud, Sertan Girgin, Sholto Douglas, Shree Pandya, Siamak Shakeri, Soham De, Ted Klimenko, Tom Hennigan, Vlad Feinberg, Wojciech Stokowiec, Yu hui Chen, Zafarali Ahmed, Zhitao Gong, Tris Warkentin, Ludovic Peran, Minh Giang, Clément Farabet, Oriol Vinyals, Jeff Dean, Koray Kavukcuoglu, Demis Hassabis, Zoubin Ghahramani, Douglas Eck, Joelle Barral, Fernando Pereira, Eli Collins, Armand Joulin, Noah Fiedel, Evan Senter, Alek Andreev, and Kathleen Kenealy. Gemma: Open models based on gemini research and technology, 2024.

[40] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[41] Wikipedia. Braille — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Braille&oldid=1240822029, 2024. [Online; accessed 27-August-2024].

[42] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing

Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. Qwen2 technical report, 2024.

[43] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training, 2023.

[44] Yu Zhang, Daniel S. Park, Wei Han, James Qin, Anmol Gulati, Joel Shor, Aren Jansen, Yuanzhong Xu, Yanping Huang, Shibo Wang, Zongwei Zhou, Bo Li, Min Ma, William Chan, Jiahui Yu, Yongqiang Wang, Liangliang Cao, Khe Chai Sim, Bhuvana Ramabhadran, Tara N. Sainath, Francoise Beaufays, Zhifeng Chen, Quoc V. Le, Chung-Cheng Chiu, Ruoming Pang, and Yonghui Wu. Bigssl: Exploring the frontier of large-scale semi-supervised learning for automatic speech recognition. *IEEE Journal of Selected Topics in Signal Processing*, 16(6):1519–1532, October 2022.